1999

The Use of Cooperative Groups in Learning How to Use New Software Programs

James R. Grunwald
Nova Southeastern University, grunwajr@mls-wels.edu

The Use of Cooperative Groups in Learning How to Use New Software Programs

by

James R. Grunwald

A Dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

School of Computer and Information Sciences
Nova Southeastern University

1999

We hereby certify that this dissertation, submitted by James R. Grunwald, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

Steven R. Terrell, Ed.D.
Chairperson of Dissertation Committee

Date

Laurie P. Dringus, Ph.D.
Dissertation Committee Member

Date

Maxine S. Cohen, Ph.D.
Dissertation Committee Member

Date

Approved:

Edward Lieblein, Ph.D.
Dean, School of Computer and Information Sciences

Date

School of Computer and Information Sciences
Nova Southeastern University

1999

An Abstract of a Dissertation Submitted to Nova Southeastern University
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

The Use of Cooperative Groups in Learning How to Use New Software Programs

by
James R. Grunwald

April 1999

Many teachers lament the fact that they often need to spend too much classroom time training students how to use a particular software program before their students can use it effectively as a tool to complete other course work. Also, after completing the training process, too many students still have not mastered the basics of how to use the program effectively. The purpose of this research was to increase understanding of how the use of cooperative groups during new software training affects both the mastery of the basics of program use, as well as the amount of time needed to learn the basics.

The students in the study were high school sophomores learning to use the dynamic geometry program, *The Geometer's Sketchpad*. These students were divided into three groups. Each student in the control group worked independently of one another through training materials at separate computers. The students in the other two groups worked through the training materials in cooperative pairs. In one experimental group, each pair shared a single computer and in the other experimental group each student of the cooperative pair had access to a separate computer. After completing the training sessions, each student in the study worked independently through an assessment activity.

The results of the research study indicated that the use of cooperative pairs while learning to use a new software program had a statistically significant effect on the amount of time needed by the students to learn the basics of using the software program. The research also revealed that the way in which the cooperative pairs shared computers made a difference. Students who went through the training process utilizing cooperative pairs with one computer per pair used significantly less time to completed the training tutorials than students who trained in cooperative pairs with a separate computer for each student.

The study also indicates that the training method used, whether individual or some form of cooperative pairs, had little or no effect on the mastery of the basics of program use. The researcher mentions several limitations of the study which may have contributed to the similarity of results observed between training methods, including the lack of difficulty of the materials being learned, the validity of the post-assessment activities used, and the small sample sizes of the groups.

It is suggested that more research in the area of using cooperative pairs during the training process needs to be completed before schools drastically change their current training formats. The one noted exception to this is that schools that currently make use of cooperative pairs during the training process may find it advantageous to have the two students in each pair share a single computer. The suggestion is made that additional research should be carried out with larger sample sizes and should concentrate on two groups, a control group, with one computer per student working through the training materials independently, and an experimental group in which students work in cooperative pairs with a single computer for each pair.

Acknowledgments

I would like to thank my dissertation chairperson, Dr. Steve Terrell, and fellow committee members, Dr. Laurie Dringus, and Dr. Maxine Cohen, for their valuable insight, direction, and encouragement throughout this research process. Without their continuing guidance, the process could not have come to fruition.

I would also like to thank the administration of Michigan Lutheran Seminary, in particular Assistant President William Zeiger and President Paul Prange, for their continuing support and encouragement during my studies and the research process. May God continue to bless the work being done at Michigan Lutheran Seminary.

Finally, I would like to thank my wife Karen, and my children Melanie, Nathan, Aaron, and Timothy, for their understanding, patience, and support during these past years.

Table of Contents

Appendices

List of Tables

Tables

Chapter I

Introduction

Problem Statement and Goal

Past research has shown that computers and electronic media can be used as an effective tool for the enhancement of the learning process in the classroom (Molnar, 1997; Hawkins, 1997). This fact is also supported by a meta-analysis of 35 computer related studies conducted by Liao (1998) in which Liao concluded that classroom teachers have plenty of research-based evidence indicating that the use of technology in instruction can result in positive outcomes in the classroom.

As the use of computers, as well as the sophistication of computer interfaces, has become more integrated into society in recent years, the need for training in the proper and efficient use of computers and software programs has become more critical (Baecker, Grudin, Buxton, & Greenberg, 1995). Baecker et al. point out that documentation and training are an essential part of the human computer interface, even more so today than in the past. It is an unrealistic expectation that a user can simply begin using a new program effectively and efficiently without first spending some time learning the basics of how the program functions and how the user can best make use of the program's features.

Likewise, in order for students to effectively use technology in the classroom, and apply it as a tool to enhance learning, they must first spend time mastering the fundamentals of the technology itself (Schatz, 1996). For example, to make good use of *The Geometer's Sketchpad* program in a geometry class, the students must first spend valuable class time learning the basics of the program. The current problem is that more class time than desired is being used by the students in class to learn the basics of the program. Also, according to Schatz and others, too many students fail to master the basics of how to use the software program during these initial training sessions.

Part of this lack of transfer between the training sessions and the actual use of the software may be due to the design of the software itself (Carroll & Mack, 1995). Studies in Human Computer Interaction (HCI), as reported by Nielsen and Mack (1994), point out that the way in which a software application program is designed can greatly affect its ease of learning and usability.

The lack of rapid learning and transfer between software training sessions and the actual use of the software is not unique to the educational arena. Two separate training studies, Olfman and Bostrom (1991), and Olfman and Mandviwalla (1992), report that less than half of the workers who attend formal software training sessions actually end up using the software later on. What is unique to the educational arena are the circumstances under which formal software training takes place and the immediacy of using the software in the classroom environment once the basics of the software program have been mastered. While much research and writing has taken place concerning improving the efficiency of software training in the business world (Shayo & Olfman, 1993, 1994), very

little, if any, has been done on how software mastery and the transfer of use can be improved within the confines of the educational arena (Bailey, 1997).

Cooperative learning, although a form of student learning which has been around for many thousands of years, has become more common in classrooms in recent years (Strommen, 1995). This is especially evident in the area of mathematics education where the use of cooperative groups has been strongly advocated by organizations such as the National Council of Teachers of Mathematics (NCTM) in their Curriculum and Evaluation Standards for School Mathematics (1989) and by authors such as Foster (1993) and Serra (1997). Researchers, such as Johnson and Johnson (1994), have found that working in small cooperative groups has a positive impact on student achievement as well as interpersonal relations, a finding which is well supported in the writings of others such as Jewett (1996), and Pace and Gardner (1997).

Even though many teachers may advocate and make regular use of cooperative groups in their classrooms (O'Connor, 1997), it may seem contradictory that these same teachers often have students learn to use new software programs independently of one another. The training in the use of new software often takes place with a single student at each computer arranged in a lab type environment, assuming, incorrectly, that students learn new software best in a passive environment (Carroll & Mack, 1995).

Prior to this study, it was unknown if the use of cooperative groups, while learning to use new software, would decrease the amount of class time needed to learn the basics of the software program, and allow more students to master the basics of using the software effectively. If the amount of time needed to learn to use the software could be reduced through the use of cooperative groups, then more time would be available to use

the software as a tool to enhance the learning of other material in the course. Also, if the percentage of students who mastered the basics of using the software increased through the use of cooperative groups, then less time would be needed for relearning the software at a later date.

The purpose, or goal, of this study was to increase understanding of how the use of cooperative groups would affect both the mastery of the basics of a software program and the amount of time needed by students to learn the basics of a software program.

Relevance and Significance

The results of recent studies show that the use of cooperative learning techniques with children foster the development of leadership skills, a sense of teamwork, and improved self-esteem (Strommen, 1995). Chiu (1995) reports that students who studied in a cooperative learning mode recalled significantly more than those who studied alone. Many authors and researchers, such as Johnson and Johnson (1989, 1994), Slavin (1990), Johnson, Johnson, and Smith (1991), Dishon and O'Leary (1994), Serra (1997), and others, have expounded upon how teachers can effectively use cooperative, or collaborative as they are sometimes called, learning techniques in the classroom.

The use of cooperative learning techniques in the classroom is not something new to the educational community (Strommen, 1995). Roschelle (1994) reports that along with the introduction of technology into the classroom, a renewed interest in the use of cooperative learning is taking place. Uslick and Walker (1994) mention that teachers feel that the computer enhances and facilitates the use of cooperative learning in the

classroom. Other authors, such as Pace and Gardner (1997) and O'Connor (1997), also report that this renewed interest in the use of cooperative groups is continuing into present day classrooms.

Much also has been written concerning the benefits of using cooperative learning and technology in the mathematics classroom in recent years. The National Council of Teachers of Mathematics (NCTM) has added much to the discussion by publishing their Curriculum and Evaluation Standards for School Mathematics (1989), Professional Standards for Teaching Mathematics (1991), Assessment Standards for School Mathematics (1995), as well as an Addenda Series to supplement the Curriculum and Evaluation Standards. Each of these publications, as well as others, supports the use of cooperative learning and the use of technology in the mathematics classroom. Research has shown that cooperative learning and the use of technology help to increase the achievement levels of students (Strommen, 1995; Pace & Gardner, 1997).

Research has also shown that the skillful and appropriate use of a variety of techniques over time, as opposed to a prolonged use of a single technique or approach to presenting material, results in higher student achievement (Eddins, Maxwell, & Stanislaus, 1994). Taylor, in his book *The Computer in the School: Tutor, Tool, Tutee* (1980), tried to expand the usefulness of the computer in education. The computer was seen as more than simply a means to present computer assisted instruction (CAI) drill and practice and tutorials to the student. It was also seen as a tool to aid the student, teacher, and workers in society, and as a tutee, or learner, that could be instructed by the teacher or student.

Other authors, such as Perelman (1992) and Papert (1993), also see the computer as a catalyst for changing the current view of school, the learning process, and the focus of education, from a passive, teacher-centered environment, to an active, discovery oriented, learner-centered environment. Many educational writers, such as Riedl (1987), Grunwald (1990), Dyrli and Kinnaman (1994), and educational publications such as the Curriculum and Evaluation Standards for School Mathematics (NCTM, 1989), now embrace this broad view of the use of technology in the classroom. As a result, more and more teachers are beginning to use technology in the classroom to enhance the curriculum, and expect their students to use it also. But, this influx of technology into the classroom has also created the issue of how to best train students in the use of software programs.

Baecker et al. (1995) point out that training is an essential part of any human computer interface. Without adequate training concerning the basic usage of a software program, the user will generally not make use of the program's features in the most expedient fashion. This concept is supported by the research of Wiedenbeck and Zila (1997) who found that open-ended exploration by the learner in learning how to use a new software package tends to be unsuccessful. Also, just because formal software training takes place is no guarantee that the software will be used once the training sessions are completed (Shayo & Olfman, 1993). Much of this research, however, has been conducted in the area of software training in the business world, which is quite different from the educational arena where students often learn to use a software program for immediate application within a particular class.

As stated previously, it was unknown if the use of cooperative groups, while learning to use a software program, would decrease the amount of class time needed to learn the basics of the software program, and allow more students to master the basics of using the software effectively. If the amount of time needed to learn to use the software could be reduced through the use of cooperative groups, then more class time would be available to use the software as a tool to enhance learning. Also, if the percentage of students who mastered the basics of using the software increased through the use of cooperative groups, then less time would be needed for relearning the software at a later date.

Although much has been written concerning the use of cooperative groups in the classroom (Saxton, 1995), and much has been written concerning effective software training techniques in the business community, research which looks specifically at the use of cooperative groups associated with the learning of new software programs in the educational arena is very sparse, a research deficiency also reported by Bailey (1997). If it could be shown, through research, that the use of cooperative groups does speed up the learning process of new software and/or increases the retention of the basics needed to use the software program effectively, then schools may need to reexamine how they train their students to use software packages.

Barriers and Issues

The fact that little, if any, research has been directed specifically at the use of cooperative groups in the training of students to use new software programs (Bailey,

1997) is not as surprising as one might first think. First, most research in the area of software training techniques has taken place within the realms of the business community. In the business world, participants in formal training sessions often come from various companies, are representative of various age groups and experience levels, and have various motives for attending the training sessions and for learning to use the new software. With such a diverse background of participants and goal expectations, individualistic training methods may well work the best (Shayo & Olfman, 1993). However, many educators point out that school environments are quite different from the business world. Students learning to use a particular software package share the same goal, learning to use the software program as a tool to conduct other work in their various classes.

Secondly, it is very difficult to measure with precision the mastery of the basics of a software package. *The Geometer's Sketchpad* program, as is the case with many software programs, is a program where usability issues such as learnability have only been tested on a summative basis. For example, according to the designer of *The Geometer's Sketchpad* program, Nick Jackiw (personal communication, August 5, 1997), no test instrument currently exists to measure how well students have mastered the basics of using *The Geometer's Sketchpad* program. All field testing of *The Geometer's Sketchpad* program has revolved around how easy it is to use and how effective the program is after students have already learned how to use it. The same is true of most training manuals and tutorials which are packaged with new software programs.

In other words, research concerning the use of such programs often begins after the students already know how to use the program. This research examines study how

students can more effectively learn the basics of how to use the program. Because of the author's extensive knowledge of *The Geometer's Sketchpad* program and its use in the classroom, a test instrument has been developed by the author to measure mastery of the basics of the software package.

Third, cooperative learning research has also focused primarily on the use of technology after the students have learned to use the program. One area, as previously mentioned, where little or no research has been done is in the use of cooperative pairs while learning how to use software programs (Bailey, 1997). The reason for the sparsity of research in this area might be attributable to the fact that many educators take for granted that the traditional passive method of the isolated one-computer-per-student model, where an individual student works at his or her own pace, is the best way to learn new software programs (Carroll & Mack, 1995).

Research Questions

As stated previously, the purpose, or goal, of this study was to increase understanding of how the use of cooperative groups, while learning to use a new software program, would affect both the mastery of the basics of the program as well as the amount of time needed by students to learn those basics. The specific research questions were:

1. Do students who cooperatively learn to use a software program, learn to use the program significantly faster than students who independently learn to use the same software program?

2. Are students who cooperatively learn to use a software program significantly more likely to master the basics of the program's use than students who independently learn to use the same software program?

Overview of the Research Study

The study was conducted at a parochial high school, with an enrollment of approximately 340 students in grades 9 through 12. The 10th-grade population of the school was selected to be subjects of the study because that is the grade level at which the software package being used in the study, *The Geometer's Sketchpad*, is first learned and then used in class as a tool by the students.

The school randomly schedules its students into various sections of geometry. The three sections of geometry, each with between 20 and 26 students, were randomly selected by the researcher to serve as either the control or one of the two experimental groups. Within the experimental group sections, the students were grouped randomly by the researcher into pairs. The study was carried out over three consecutive days in the school's main computer lab during each student's regular 50 minute geometry class period.

During the training sessions on the first two days of the study, the control group participants worked through the tutorial training materials individually at separate computers. The first experimental group section had paired students working together cooperatively, with each pair at a single computer. The second experimental group section had paired students working together cooperatively, but with each student in the

pair having their own computer. During the third day of the study, each student in all three groups worked individually at separate computers to complete an assessment tool to determine what they had learned during the previous two sessions.

At the conclusion of the experiment, the amount of time needed to complete the tutorials and assessment activities, as well as the number of students who successfully mastered the basics of the software package, were examined. An examination of the data, as detailed later in this report, revealed whether or not the use of cooperative pairs during the training process had any effect on the amount of time needed by the students to successfully learn to use the software package, and if the use of cooperative pairs had any effect on the percentage of students who successfully learned to use the basics of the software package.

Limitations and Delimitations

There are several limitations and delimitations which may affect the generalizability of this study to other populations. These limitations and delimitations are listed in the following paragraphs.

Limitations

The following limitations to the investigation are noted:

1. Students whose parents withheld parental permission did not participate in the study. This could have adversely affected the composition of the population.

2. The school in which the study was conducted was a parochial high school, supported financially by the Wisconsin Evangelical Lutheran Synod (WELS). The school limits entrance based on various criteria, one of which is a satisfactory score on a mathematics screening test. The school further limits entrance to those students who state a willingness to consider careers as pastors or teachers in the WELS's churches or schools. Typically, between 40 and 50 percent of the high school's graduates continue their studies at the synod's college of ministry, Martin Luther College, in New Ulm, Minnesota. Almost 100 percent of the remaining graduates enter other secular colleges and universities (P. T. Prange, Personal communication, May 22, 1998).

The high school also provides dormitories for approximately two-thirds of its students, with the remaining students commuting daily from home. The dormitory students have mandatory supervised study periods each evening during which time they have access to the school's networked computer lab. Commuting students may also use the computer facilities each evening (S. Post, Personal communication, May 22, 1998).

Because of the unique purpose, environment, and student body of this high school, even though it has a traditional four year mathematics sequence of algebra I, geometry, algebra II - trigonometry, and pre-calculus, generalizations to other high school populations may be limited.

Delimitations

The following delimitations to the investigation are noted:

1. Only those students enrolled in geometry at the high school at the beginning of the school year participated in the study.

2. Students who were retaking geometry because they failed it the previous year did not participate in the study.

3. Students with prior experience in the use of *The Geometer's Sketchpad* program did not participate in the study.

4. Students who participated in the English as a Second Language (ESL) program were excluded from the study.

5. Foreign exchange students, where English is not the national language, were excluded from the study.

6. Only one software program, *The Geometer's Sketchpad*, was used in this study. Generalizations to other software programs may be limited, even within this high school.

Definition of Terms

Basics - Basics, as it relates to this study, refers to the "fundamentals" or beginning knowledge set someone needs to accomplish a given task (Merriam-Webster's Collegiate Dictionary, 1995). The term is often found in this study in conjunction with the phrase "basics of the software program," referring to the beginning or fundamental tasks one needs to be able to perform in order to use the software program effectively as a tool to accomplish other, more complex, tasks.

Mastery - As used in this study, mastery refers to having obtained the skill or knowledge that makes one a master of a subject (Merriam-Webster's Collegiate Dictionary, 1995). This study often uses the phrase "mastery of the basics of the software program," referring to having learned the basics of how to use the program well enough

to be able to successfully use the program as a tool to accomplish other more complex tasks successfully.

Cooperative Learning - Cooperative learning refers to the instructional use of small groups of students who work together on learning activities to maximize their own and each other's learning (Johnson & Johnson, 1989; Serra, 1997).

Cooperative Pairs - Cooperative pairs refer to one particular way in which cooperative groups can be formed. Such groups consist of two students working together as a pair on a learning activity (Johnson & Johnson, 1989).

Summary

Past research has shown that computers and electronic media can be used as an effective tool for the enhancement of the learning process in the classroom (Molnar, 1997; Hawkins, 1997). As the use of computers, as well as the sophistication of computer interfaces, has become more integrated into society in recent years, the need for training in the proper and efficient use of computers and software programs has become more critical (Baecker, Grudin, Buxton, & Greenberg, 1995). It is an unrealistic expectation that a user can simply begin using a new program effectively and efficiently without first spending some time learning the basics of how the program functions and how the user can best make use of the program's features.

Likewise, in order for students to effectively use technology in the classroom, and apply it as a tool to enhance learning, they must first spend time mastering the fundamentals of the technology itself. The problem which led to this research study was

that more class time than desired was being used by the students in class to learn the basics of the program and too many students fail to master the basics of how to use the software program during these initial training sessions (Schatz, 1996).

Cooperative learning has become more common in classrooms in recent years (Strommen, 1995). Researchers, such as Johnson and Johnson (1994), have found that working in small cooperative groups has a positive impact on student achievement as well as interpersonal relations.

It was unknown if the use of cooperative groups, while learning to use new software, would decrease the amount of class time needed to learn the basics of the software program and allow more students to master the basics of using the software effectively during the initial training sessions.

The purpose, or goal, of this study was to increase understanding of how the use of cooperative groups while learning to use a new software program would affect both the mastery of the basics of the software program and the amount of time needed by students to learn the basics of the software program.

Chapter II

Review of Literature

This chapter reviews the current literature and research as it relates to the use of cooperative groups to enhance the effectiveness of learning to use new software programs. The chapter begins by examining the use of cooperative learning techniques in the classroom. Next, the use of technology in the classroom is examined, including its use in cooperative learning environments. The next section examines software training in light of available research, both in the business world and the educational arena. Section four summarizes what is known and unknown about the use of cooperative learning during training sessions aimed at learning to use new software programs. The chapter concludes by pointing out the contributions which this study will make to the field of software training as well as provide a basis for the research design which is developed in chapter three.

Cooperative Learning

Cooperative learning is defined by Johnson and Johnson (1989) as the instructional use of small groups so that students work together to maximize their own and each other's learning. Serra (1997) provides a similar definition stating that

cooperative learning, or cooperative small group instruction, refers to classroom techniques in which students work together in small groups on learning activities and receive recognition based on the group's performance. This section of the chapter examines research findings as they pertain to the use of cooperative learning in education, various cooperative learning techniques, cooperative learning in the mathematics classroom, the use of technology to support cooperative learning, and the movement towards an increased use of cooperative learning in the mathematics classroom.

Cooperative Learning in Education

The use of cooperative learning techniques in the classroom is not something new to the educational community, in fact, the use of cooperative learning has been around for thousands of years (Strommen, 1995). Strommen mentions that study after study has indicated that cooperative learning consistently yields superior results in almost every area when compared to other learning techniques. Roschelle (1994) reports that along with the introduction of technology into the classroom, a renewed interest in the use of cooperative learning is taking place. Uslick and Walker (1994) mention that teachers feel that the computer enhances and facilitates the use of cooperative learning in the classroom. Other authors, such as Pace and Gardner (1997) and O'Connor (1997), report that this renewed interest in the use of cooperative groups is continuing into today's classrooms.

The results of recent studies show that the use of cooperative learning techniques with children foster the development of leadership skills, a sense of teamwork, and improved self-esteem (Strommen, 1995). Chiu (1995) reports that students who studied

in a cooperative learning mode recalled significantly more than those who studied alone. Many authors and researchers, such as Johnson and Johnson (1989, 1994), Slavin (1990), Johnson, Johnson, and Smith (1991), Dishon and O'Leary (1994), Serra (1997), and others, have expounded upon how teachers can effectively use cooperative, or collaborative as they are sometimes called, learning techniques in the classroom.

Dishon and O'Leary (1994) point out that one of the primary purposes of using cooperative groups is to teach students social skills, skills which are needed not only to complete the current task successfully, but which are needed to make working in cooperative groups an enjoyable experience for all students in the group, a point echoed by Jewett (1996). Many occupations in society require that employees are able to work in small groups with fellow employees in a constructive, problem-solving fashion.

Johnson and Johnson (1989), after conducting extensive research, have identified five essential elements of effective cooperative learning structures. First, positive interdependence is essential. The cooperative learning experience must be designed so that all participants contribute to the collaborative task of the group and each member of the group feels needed. Secondly, face-to-face promotive interaction needs to be practiced. Group members, in face-to-face gatherings, promote each other's learning by helping, encouraging, and supporting one another during the cooperative learning experience. Third, individual accountability is required. Students, while working together to complete the cooperative task, still need to be held accountable for, and assessed on, their own individual learning, as well as accountable for their role in the group effort. Fourth, students need to develop interpersonal and small group skills in order to be effective members of a cooperative group. They need to learn how to work

together as a unit, developing positive social interaction skills, which are necessary in many occupations in society. Finally, group processing time, time for the students to evaluate how well they are working together as a group, must be provided.

In summary, cooperative learning techniques are not something new to the educational community, but interest in and the use of cooperative groups has increased in recent years, partially due to the use of technology. Many researchers have found that the use of cooperative groups consistently yields superior results when compared to other learning techniques, and leads to the development of leadership skills, improved social skills, a sense of teamwork, and improved self-esteem. Also, five essential elements of effective cooperative learning structures have been identified.

Various Cooperative Learning Techniques

Much research has been done concerning the use of cooperative learning as a teaching strategy for the classroom. This section reports on some of the findings which have surfaced as a result of this research. Group sizes, various methods used to assign students to a group, and student roles within a group are discussed.

Cooperative learning can be used in a variety of configurations. Groups sizes can vary anywhere from two to six members, depending on the type of activity being done. Research, however, has found that groups of four or five work best for most cooperative learning situations (Serra, 1997). However, Serra also points out that groups of two, or cooperative pairs as they are often referred to, seem to work best when doing computer activities, a finding also supported by O'Malley (1992). Such pairs allow for one student to type at the keyboard while the other verbally reads the instructions of the activity, or

both students can be working at adjacent computers on the same activity, comparing computer screens and helping one another work through the activity.

Students can be assigned to groups in various ways, ranging from random assignment, such as a picking numbers out of a hat, to student selected groups, to direct placement by the teacher. Foster (1993), Serra (1997), and others, suggest that the best way to arrange students in order to create groups that will be the most beneficial and productive for all students involved, and at the same time will avoid many of the negative social problems connected with student selected groups, is for the teacher to create the groups based upon each student's ability in the particular subject area being studied. With groups of four, this is accomplished by the teacher placing the highest, lowest, and two middle ability students in the first group, and then placing the second highest, second lowest, and the next two middle ability students in the second group, etc. If the teacher does not know the abilities of the students, then a random assignment procedure is a good alternative.

Johnson, Johnson, and Smith (1991) have found that the greatest problem with group composition usually occurs with student selected groups. Such groups, often consisting of students who are close friends and homogeneous in ability, tend to spend less time on-task than teacher selected heterogeneous groups. This finding is supported by others, such as Jewett (1996), who recommends only teacher formed groups. Many authors and educators also suggest forming new groups every six to nine weeks so that students become accustomed to working with other students.

Each student in the group should have a specific role assigned to them, and they need to be instructed concerning the function of their role (Johnson & Johnson, 1989).

These roles can have various names such as taskmaster, materials handler, reader, recorder, checker, encourager, gatekeeper, cheerleader, etc. If each member has a specific, unique function in the group, it is easier for the group members to remain on task and contribute to the overall success of the group. These roles should be rotated among the groups members whenever a new cooperative learning activity begins (Foster, 1993).

In summary, research has found that cooperative groups of four or five members each work best for most cooperative learning situations, but cooperative pairs seem to work best when working on computer activities. Groups can be created in various ways, but research has found that student selected groups are usually the least productive and teacher assigned heterogeneous groups, based on student ability, are the most productive and beneficial to all participants. Groups should be changed occasionally so students learn to work with other students, and each student within a group should have a specific role assigned to them in order to create a sense of responsibility to the group.

Cooperative Learning in the Mathematics Classroom

Cooperative learning in the mathematics classroom has become more popular in recent years. This section examines who is behind the surge toward the use of cooperative learning in the mathematics classroom. Also, various points of view concerning the extent to which cooperative learning should be employed are examined.

Much has been written concerning the benefits of using cooperative learning and technology in the mathematics classroom in recent years. The National Council of Teachers of Mathematics (NCTM) has added much to the discussion by publishing their

Curriculum and Evaluation Standards for School Mathematics (1989), Professional Standards for Teaching Mathematics (1991), Assessment Standards for School Mathematics (1995), as well as an Addenda Series to supplement the Curriculum and Evaluation Standards. Each of these publications, as well as others, support the use of cooperative learning and the use of technology in the mathematics classroom. Research has shown that cooperative learning and the use of technology help to increase the achievement levels of most students (Pace & Gardner, 1997; Strommen, 1995).

Research has also shown that educators should not rely solely on one method of instruction, but that a variety of techniques should be employed (Eddins, Maxwell, & Stanislaus, 1994). This holds true for the use of cooperative groups in the mathematics classroom as well. However, some educators, such as Dubinsky and Schwingendorf (1997), maintain that cooperative learning is the right context for a mathematics course, especially when the pedagogical approach to the course is based on the constructivist theory of learning mathematics. This constructivist view, that students need to construct or discover knowledge in order to really learn it, is shared by other proponents of cooperative learning in the mathematics classroom, such as Michael Serra (1997) in his book *Discovering Geometry* by Key Curriculum Press. Vince O'Connor (1997), on the other hand, cautions that cooperative groups should not be looked upon as the best, or only, method of instruction in all situations. O'Connor goes on to cite many examples of ways in which various teachers at various grade levels are making use of cooperative learning in mathematics classrooms.

To summarize, the NCTM is seen as a major force behind the mathematics reform movement, which embraces the use of cooperative learning techniques for the

mathematics classroom. Some educators go even so far as to say that cooperative learning techniques should be the main stay for all mathematics instruction, especially those who are strong supporters of the constructivist view of mathematics learning. Others caution that cooperative learning is only another technique in the arsenal of tools which mathematics teachers can make use of in the classroom.

Use of Technology to Support Cooperative Learning

The use of technology is being seen by many as the ideal tool to help support cooperative learning. This section examines recent research and literature concerning this supportive role of technology. The discussion points out three distinct roles which the computer can play with respect to collaboration. Finally, the acquisition of problem solving skills is discussed in light of the use of technology and cooperative learning strategies.

Many researchers and educators believe that technology is the best tool to bring cooperative learning into the classroom (Strommen, 1995; Tomlinson & Henderson, 1995). Strommen states that the technology revolution has given cooperative learning an even stronger imperative than it had before. The computer is beginning to be viewed as a catalyst and tool which will increase the use of cooperative learning among teachers. Silverman (1995) reports that his investigations into the use of group collaboration tools have revealed that students perform best with a combination of minimal instruction and a constructivist environment of learning, an environment where computers are used as tools to support collaborative learning.

McMahon (1990) reports that educators often endorse the use of computers by groups of students, but then assess them individually. This is consistent with the third essential element of cooperative learning mentioned earlier by Johnson and Johnson (1989), the fact that students need to be able to demonstrate individual accountability even when working in cooperative groups. According to McMahon, there is abundant evidence that the use of computers in groups leads to positive educational gains, and that many educators take it for granted that children should work in groups. Too few educators pay enough attention to the size and composition of groups when working with computers. McMahon reported on a 1989 study which found that most children preferred to work in pairs when working with computers, although some children preferred to work alone.

O'Malley (1992) reports that computers are increasingly perceived by educators as an effective tool for enhancing cooperative learning in the classroom. Several studies indicate that the use of computers increases the level of interaction between students and encourages them to work cooperatively. O'Malley presents three distinct roles which the computer can play with respect to collaboration. The first role is "learning around the computer." In this situation the computer is used to display or reflect the results of some joint activity. In this case, the software being used is often designed for individual use, requiring students to take turns to use it. The second role is "learning through the computer." In this scenario, computers are used by students to facilitate communication between groups or pairs of students engaged in joint activities. The third role is that of "learning mediated via the computer." In this view, the computer is used as a tool which

augments collaborative learning by supporting joint activities as well as communication between a pair or group of students.

With the recent emphasis placed on the teaching of problem solving skills by the NCTM (1991) and others, emphasizing that students need to learn how to model and work on real-world problems, some educators and researchers have turned to cooperative learning techniques. Natasi, Battista, and Clements (1990) found that working on solving problems in cooperative groups forced participants to defend their position to others in the group. It was found that groups of students which were encouraged to disagree and challenge the thinking of others within their groups scored higher on subsequent achievement indicators than those in groups where consensus and the avoidance of disagreement were encouraged. Natasi et al. concluded that cooperative learning is an excellent strategy, as part of a constructivist design of learning, for developing the thinking skills necessary for productive problem solving. Denning and Smith (1995) similarly report that the cooperative learning approach, along with problem-based learning strategies and technology, are a powerful tool for teaching problem solving skills, especially to populations that are at risk of academic failure.

In summary, many educators and researchers now view the computer as a tool which will enhance the use of cooperative learning in the classroom. In fact, some educators now take it for granted that students should be using computers and working in groups as a normal part of their learning environment. The use of technology in group work is sometimes broken into three levels of use; "learning around the computer", "learning with the computer", and "learning mediated via the computer." Researchers

have also discovered that the use of computers along with cooperative learning strategies is an excellent way for students to develop problem solving skills.

Use of Technology With Cooperative Learning in the Mathematics Classroom

Many educators, researchers, and authors now point to the use of technology as a tool to be used in conjunction with cooperative learning strategies in the mathematics classroom. As an example, Uslick and Walker (1994) reported on the Lighthouse Education Enhancement Project, a four-year project designed as a collaborative effort between urban and suburban school districts to formulate plans to implement mathematics reform as presented by the NCTM (1989, 1991). As the teachers involved worked toward their goal of mathematics reform, they soon came to realize that technology was an important ingredient needed to meet the educational reforms, along with cooperative learning strategies. They began to view the computer as a tool to enhance cooperative learning. With the help of technology and cooperative learning, the teachers began to spend more time asking questions, looking for patterns, and engaging in mathematical thinking. As more and more mathematics teachers embrace discovery learning, as associated with constructionist theories of learning, they are encouraging their students to work in cooperative groups, where computers are being used as a tool of discovery.

Summary of Cooperative Learning Usage in Education

Cooperative learning is often defined as the instructional use of small groups of students working together on a learning activity. Researchers have consistently found

that the use of cooperative groups in the classroom yields superior results when compared to other learning techniques, as well as leads to the development of leadership and social skills, improved self-esteem, and a sense of teamwork among the students.

Research has determined that the format and composition of the cooperative group is important to the overall productivity of the group. Group sizes of four to five members appears to work best for most activities, however, cooperative pairs seem to work best when computer related activities are involved. Although groups can be selected in various ways, teacher assigned heterogeneous groups, based on student ability, have been found to be the most productive, while student selected groups are often least productive. It has also been determined that students often perform better in a group situation if each member of the group has been assigned a particular responsibility or role to fulfill.

While cooperative learning techniques are not something new to the educational community, a recent increased interest in their use has been noted, an increase which some educators attribute to the increased use of technology in the classroom. Many educators and researchers now view the computer as a tool which will enhance the use of cooperative learning in the classroom.

This increased use of cooperative learning techniques and technology has been very apparent in the area of mathematics education. Many educational writers attribute this recent trend to the mathematics reform movement of the NCTM and the constructivist view of learning mathematics.

Technology in Education

The use of computer technology in education is a relatively recent addition to the tools available to teachers (Molnar, 1997). This section of the chapter examines the recent literature and available research findings as they pertain to the early uses of technology in education, with a look at their use in the mathematics classroom in particular. Also examined, are the efforts aimed at refocusing the role of computers in education. Next, attention is directed towards current uses of technology in the mathematics classroom, including the emergence of dynamic graphing software. Finally, the section examines some current trends concerning the use of computers in education and what educators and researchers have to say concerning these trends.

Early Use of Technology in the Classroom

In order to fully appreciate the impact that technology has had on education, a quick review of the early use of technology in education is necessary. The uses of early computers in education, the type of educational software which was used, as well as two separate computer revolutions will are discussed in the following paragraphs.

According to Molnar (1997), the history of computers in education has been relatively brief. The first operational computer, the Mark I, was put into service in 1944 at Harvard, and was soon followed by the Electronic Numerical Integrator and Calculator (ENIAC) at the University of Pennsylvania in 1946. The first computers in education were primarily used as number crunching machines to aid in the solving of complex calculations in the fields of mathematics, science, and engineering (Withrow, 1997).

In 1959, the first large-scale educational project for the use of computers, Programmed Learning Automated Teaching Operations (PLATO), was begun at the University of Illinois by Donald Bitier. The PLATO project served undergraduate education as well as elementary school reading programs by connecting students via several thousand terminals to a main frame computer (Molnar, 1997).

The first educational computer revolution, as Dockterman (1997) refers to it, occurred in the 1960's. In 1963, John Kemeny and Thomas Kurtz were instrumental in converting the use of computers in education from a research tool to an academic one. Time-sharing systems were developed which allowed students at remote terminals to share in the use of a mainframe system. During this same time period Kemeny and Kurtz developed the easy to use high-level programing language BASIC (Beginners All-Purpose Symbolic Instruction Code). The use of BASIC spread quickly and was used to develop computer based instruction (CBI) materials for a variety of subjects and educational levels. Molnar (1997) reports that CBI materials were looked at as a way to free students from the group-paced instruction of the typical classroom. Students would be able to progress at their own pace, and along individualized paths, as the computer rapidly provided them with important feedback. Most CBI programs during this era made use of drill and practice techniques to obtain mastery of the concepts being presented. According to Dockterman, this first computer revolution did not catch on for several reasons, including the fact that the terminals were too expensive and finicky.

The second educational computer revolution began in the late 1970's or early 1980's and was spawned by the introduction of the microcomputer (Dockterman, 1997; Molnar, 1997). The computer was finally freed from its connection to a mainframe. It

now was available to anyone who wanted one, and could afford it. Computers soon began to move into businesses, homes, and schools, however, the software programs used in education, especially in the area of mathematics, were still predominately of the drill and practice variety.

In summary, the use of computers in education has had a relatively brief history. The first operational mainframe computers were used primarily as number crunching machines to aid in solving complex calculations in the fields of mathematics, science, and engineering. The first educational computer revolution occurred in the 1960's when time-sharing systems, with remote terminals for student use, were developed and located on a few university campuses. During this same time period, the BASIC programming language was developed and was often used to develop drill and practice computer based instruction (CBI) materials. These early CBI materials enabled students to progress at their own pace along individualized paths of learning, but were not widely used. A second computer revolution began in the late 1970's or early 1980's when the stand-alone microcomputer became available to schools. However, drill and practice techniques still dominated the available educational software.

Early Use of Technology in the Mathematics Classroom

The early use of computers in education was most heavily concentrated in the area of mathematics education. This section examines how computers were often used in mathematics classrooms as well as what researchers have to say concerning their use.

Mathematics, a discipline where drill and practice activities were common even before the advent of computers, appeared to many to be a natural place to introduce the

use of technology into the classroom. Early studies into the effectiveness of using technology to improve mathematics achievement usually focused on a comparison between traditional paper drill and practice activities and computer assisted instruction (CAI) drill and practice software and tutorials. These studies resulted in mixed findings. Leigh et al.'s 1984 study (as cited in Casey, 1987) found that there was no significant gain in student achievement between those who used computers for drill and practice and those who didn't. Yet, Casey cites a 1984 study by Kulik et al. who conducted a meta-analysis of 29 studies comparing the use of CAI drill and practice and tutorials with traditional paper forms of drill and practice and found that CAI drill and practice and tutorials increased student achievement by 0.48 standard deviations. Another down-side to early CAI software, besides the mixed reviews concerning its effectiveness, was that it was found to be behavioristic in that it turned learning into a systematic, controlled form of work (Casey, 1987).

While many research studies in the 1980's pointed to positive results from the use of CAI drill and practice software, there were those who believed that "it was a shame to use such a marvelous device as a computer for 'mere' drill and practice, when it could be used for seemingly more creative activities" (Salisbury, 1985, p. 1). Strommen (1995) echoed similar concerns in reporting that educational experts estimate that the predominant use of technology in schools, as of 1995, still remains automated drill and practice learning.

In review, the early use of computers in education was most heavily concentrated in the area of mathematics education. The software programs used were primarily of the drill and practice variety because drill and practice was a common method of review used

in the mathematics classroom even before the advent of computers. Various research studies have resulted in conflicting findings concerning the effectiveness of drill and practice software over traditional paper and pencil drill and practice activities, yet it is estimated that drill and practice software still remains the predominant form of software found in the mathematics classrooms as recently as 1995.

Efforts to Refocus the Role of Computers in Education

In an attempt to move away from using computers simply for drill and practice or tutorial activities, or even as an object to be studied in and of itself, several educators have tried to expand the view of the usefulness of computers in education. During the past two decades, there have been several efforts aimed at refocusing the use of computers in education. This section examines the calls for reform, as well as how those calling for reform envisioned the computer being used effectively in the classroom.

Taylor, in his book *The Computer in the School: Tutor, Tool, Tutee* (1980), tried to lead others to expand the usefulness of computers in education. The computer was seen by Taylor as more than simply a means to present CAI drill and practice and tutorials to students. It was also seen as a tool to aid students, teachers, and workers in society, and as a tutee, or learner, that could be instructed by the teacher or student.

Other authors, such as Perelman (1992) and Papert (1993), also saw computers as a catalyst for changing the current view of school, the learning process, and the focus of education, from a passive, teacher-centered environment, to an active, discovery oriented, learner-centered environment. Many educational writers, such as Riedl (1987), Grunwald (1990), Dyrli and Kinnaman (1994, 1995), and educational publications such as those of

the NCTM (1989, 1991, 1995), now embrace this broad view of the use of technology in the classroom. In this view, the computer is a tool which is only as useful as the decisions which educators and students make concerning how to use it (McSweeny, 1997). As a result, more and more teachers are beginning to use technology in the classroom to enhance the curriculum, and expect their students to use it also. But, this influx of technology into the classroom has also created the issue of how to best train students in the use of new software programs.

Some authors, such as Gates (1997) in his book *The Road Ahead*, maintain that so far, education remains relatively unchanged by the introduction of technology into the classroom. In fact, some educators and authors, such as Griest (1996), see the introduction of the computer into the classroom as an obstacle to the improvement of education, rather than a catalyst for change, a view shared by Papert (1993) in his book *The Children's Machine: Rethinking School in the Age of the Computer*.

Papert (1993) and Griest (1996) both mention that during the influx of computers into educational institutions in the 1980's, most computers in schools became centralized in a single room or computer lab in order to facilitate learning how to use the computer. However, this clustering of the computers into a single area greatly inhibited the integration of computers into the curriculum by teachers. In fact, Papert and Griest report that computer labs actually prevented many teachers from ever coming in contact with a computer, after all, "computer skills" were usually taught by a specialized computer science teacher, not the regular classroom teacher. Computers became a new subject to study, rather than a tool to be used in all subject areas to enhance the learning process. Unfortunately, as recently as 1996, it was reported that 54 percent of schools in one study

still indicated that they were keeping computers in labs rather than individual classrooms, an environment which many educators feel reinforces using computers outside of any relevant learning context (Kennedy, 1996).

To summarize, many educators feel that the computer should be used for more than just running drill and practice or tutorial software, or as an object to be studied in a computer science class. Many reformers see the computer as a catalyst for changing the current view of school, the learning process, and the focus of education, from a passive, teacher-centered environment, to an active, discovery oriented, learner-centered environment. These educators are calling for the computer to be viewed and used as a tool in all subject areas by teachers and students alike to enhance the learning process. These same educators are asking that computers be moved out of computer labs and into individual classrooms where they can be used as a tool by all. But even if this happens, the question still remains of how to best train students in the use of new software programs.

Current Use of Technology in Mathematics

This section examines the current use of technology in mathematics instruction as it is supported through recent research. The effects of the recent mathematics reform movement on the use of computers in mathematics classrooms are explored. Recent research studies, comparing CAI instruction to traditional instruction methods, are examined, along with the shifting focus in the mathematics classroom from memorization of rules towards problem solving of real-world problems, and how this shift relates to the role of technology.

Since the start of the mathematics reform movement in the late 1980's or early 1990's, as characterized by the collective works of the National Council of Teachers of Mathematics (NCTM, 1989, 1991) and others, mathematics teachers have been moving toward a new view of how mathematics is best learned by students and, therefore, how mathematics should be taught. This view of how mathematics is best learned sees the student not as a passive recipient of knowledge, but as an active participant involved in social interaction, constructing, discovering, conjecturing, and analyzing mathematical concepts and skills (DeCorte, 1992). The NCTM calls for mathematics teachers to use tables, graphs, algebraic and geometric models, and technology as tools to interpret real-world situations, expressions, equations, and problems.

At the same time, several studies in the early 1990's, comparing CAI to more traditional instruction methods in the mathematics classroom, found that CAI did not lead to a significant increase in student achievement (Alexander, 1991; Barnes, 1991). However, they did show that CAI was as effective as traditional teaching methods and that CAI helped students feel more positive about themselves. Apparently, even if students didn't learn more by using CAI methods, at least they were more positive about their learning experience.

Britt, Eurich-Fulcer, and Schofield (1994) examined the apparent paradox that students preferred to work with CAI materials, even when they felt the teacher did a better job of explaining things. In a two-year qualitative study, Britt et al. observed students as they worked with an artificial CAI geometry tutor program designed to assist them in the learning of geometry. They found that even though the students felt that the classroom teacher provided them with a better explanation of geometry, they still

preferred to work harder with the CAI tutor than in the traditional classroom teaching format. The researchers believe that there are two primary reasons for this. First, the CAI tutor did not eliminate the teacher, but simply changed the teacher's role from the main provider of information to a resource person who could provide individualized help. Secondly, the students felt less intimidated or less embarrassed when they could use the CAI tutor to review a concept or to ask for a further explanation of a term than having to ask the teacher in front of the other students.

Another area of mathematics receiving additional attention in recent years is problem-solving. The NCTM and others are pointing out that students need to learn how to model and work on real-world problems. Technology, along with the appropriate software, can be used to facilitate problem-solving by students (Enderson, 1997; Widmer & Sheffield, 1998). Dugdale, LeGare, Matthews, and Ju (1998) point out that even though technology is recognized as a useful tool for problem solving, very few students spontaneously use technology effectively. Good problem solvers need to recognize when the use of a technology tool is appropriate, and which technology tools can best be used in a particular situation. Dugdale et al. report that their studies show that appropriate spontaneous technology selection and use can be taught to students.

In summary, the recent mathematics reform movement has caused many mathematics teachers to move toward a view of mathematics instruction which is student-centered, where the student is an active participant involved in social interaction, constructing, discovering, conjecturing, and analyzing mathematical concepts. The NCTM calls for technology as one of the tools which teachers should use to help meet this vision of a student-centered constructivist classroom. However, some studies which

compared CAI to more traditional methods of instruction in the classroom did not find CAI to lead to greater learning, but did find that it lead to a more positive learning experience. In addition, one study found that even though students preferred the teacher's explanation of geometry concepts over an artificial CAI geometry tutor's, the students still preferred to work with the tutor because of the anonymity involved and the change in the teacher's role from main provider of information to a resource person. The focus of learning in the mathematics classroom has also shifted away from the memorization of rules to one of problem solving skills. To aid in this effort, research has shown that technology, along with the appropriate software, can be used to help facilitate problem-solving by students.

The Emergence of Dynamic Graphing Software

A good example of a software program which has been adopted by many schools as a tool to enhance learning is dynamic graphing software. Dynamic graphing software tools can be used in mathematics classes to explore various concepts connected with the study of geometry. Through their use, students are encouraged to explore, construct, investigate, conjecture, and analyze various geometric concepts. In this section, the emergence of dynamic graphing software, its current use in the classroom, and the need for teachers to be trained in how to use the software, is examined.

One successful example of using the computer as a tool to enhance the learning environment can be taken from the area of mathematics and the emergence of geometrical drawing software. Although many mathematics educators view the emergence of geometrical drawing software as a relatively recent occurrence, the first such software

actually appeared in high schools and elementary schools in the late 1970's or early 1980's in the form of the programming language Logo. Papert, in his book *Mindstorms* (1980), "describes how Logo enables children to enter 'mathland', a place where they can explore sophisticated, advanced mathematical concepts such as differential geometry, but in terms and ways that they understand and enjoy" (Coburn et al., 1985, p. 67). Ploeger (1984) writes that "However, the research indicates that the value of Logo is more anecdotal and imagined than demonstrated" (p. 269). Rather than being used as a tool for exploration in geometry, Logo was often used as a beginning level programming language for elementary school students. However, some universities and colleges did use a form of Logo called "turtle geometry" to learn about geometric concepts such as recursion, nested triangles, curves, spirals, and vectors (Abelson & diSessa, 1982).

In the early 1990's the current generation of geometrical drawing software emerged, *The Geometer's Sketchpad* in 1990, *Cabri: The Interactive Notebook* in 1992, and *The Geometry Inventor* in 1992. Each of these software packages is making a remarkably rapid entry into high schools and middle schools. This new brand of software can allow the user to make constructions, measure angles, and perform computations worthy of further proof. The concept of "continuous change" is what sets these software packages apart from simple worksheets or textbook examples (Cuoco, Goldenberg, & Mark, 1994).

Although little, if any, research has been done on the effect that these geometrical drawing programs have on student achievement, mathematics teachers are excited about the possibilities in this area. As stated by Hoyles and Noss (1994), "... the fact that this kind of activity might somehow lead to a more radical and widespread understanding of

geometry is just too tantalizing to ignore" (p. 716). Research has shown that the skillful and appropriate use of a variety of techniques over time, as opposed to a prolonged use of a single technique or approach to presenting material, results in higher student achievement (Eddins, Maxwell, & Stanislaus, 1994).

Because teachers often teach in a similar fashion to the way they were taught (Rasmussen, 1996), it is important for new teachers to experience the use of technology as a learning tool before they begin to teach others. Researchers Tayeh and Pokay (1994) did just that by developing a project for preservice teachers. The goal of the project was for the preservice teachers to experience first hand how their own understandings of mathematical concepts could be enhanced through the use of technology. The main technology tool used in the project was the program *The Geometer's Sketchpad,* which the preservice teachers used to form conjectures and theories while they worked in cooperative groups. The preservice teachers reported a greater appreciation, understanding, and feeling of comfort with the use of technology in mathematics education as a result of the project.

In summary, the use of dynamic graphing software is one example of how the computer can be used as a tool to enhance the learning of a particular subject. The current generation of geometrical drawing software emerged in the early 1990's, and can be used as a tool by students as they explore various concepts connected with the study of geometry. These new software programs have been rapidly adopted for use in high schools and middle schools even though little research, if any, has been done on the effect that these geometrical drawing programs have on student achievement. Because teachers tend to teach in the same manner that they were taught, it is also important for teachers to

experience first-hand these new tools of learning before they employ them in the classroom.

Current Trends of Computers in Education

A recent trend in the use of technology in education has been the use of hypermedia, which is defined by Liao (1998) as a combination between hypertext and multimedia. Hypermedia documents allow students to click on various objects in a computerized document, allowing them to rapidly move to a new location in the document or activate other mediums such as sound, still pictures, or video, in order to obtain additional information on the current topic. Liao did a meta-analysis of 35 recent studies comparing the effects of hypermedia instruction and traditional instruction on students' achievements. It is interesting to note that 57% of the studies employed individual instruction, 22% used small-group instruction, and only 11% used large-group instruction methods. The results of the meta-analysis indicated that the effects of using hypermedia instruction were positive when compared to traditional instruction methods. This once again provides educators with researched-based evidence that the use of technology in the classroom does produce positive results.

Another strong trend in education is to move computers out of labs and back into individual classrooms where both students and teachers can use them as tools of learning and instruction (November, 1997). The decentralization of computers back into individual classrooms will allow teachers to use them as a tool to present new material to the entire class. Likewise, students will be able to use one of the classroom computers to do research, work on assignments, or work through a tutorial without having to move to

the computer lab. In order for teachers, and students, to use this tool properly, training needs to take place (Holzberg, 1998).

Gates (1995) and other visionaries see the biggest impact of technology on educational institutions occurring soon, as the computer is finally recognized as a tool to be used to enhance the educational process, rather than a subject to be studied. The computer will become a tool through which students will have unparalleled access to instant information, a tool which will increase individualized learning and learning on demand (Hawkins, 1997). Along with this change, the role of the teacher will change from an information source to a facilitator and guide along the road of learning.

Bork (1997), in analyzing the past and current problems in education, promotes a future view of technology in education where technology is used primarily for individualistic instruction. Bork calls for the creation of highly interactive computer-based courses. Such courses would replace traditional courses as they are currently known because each individual student could start and progress through the course at his or her own rate. Such courses could be offered at traditional schools or via distance education formats.

One example of individualized computer assisted learning is taking place at the Grossmont College Computer Skills Enhancement Lab (CCLab) in El Cajon, California. The CCLab is designed to be an open-entry open-exit computerized lab where students can come and individually work their way through computerized courses. Students can pick and choose which courses and topics they wish to study, and may take computerized tests to access their progress and receive certificates of achievement (Smith & Tarkow, 1998).

Another recent, and growing trend in education is the use of computers for distance education courses (Kelly & Leckbee, 1998). It is expected that these online courses will increase the efficiency of education by providing education on demand and to remote locations (Withrow, 1997). In fact, Withrow goes so far as to predict that if the present trend in digital technologies continues, the impact of technology on education will be comparable to the impact produced by the invention of the printing press.

No matter what the future trends in the educational use of technology may be, there still remains the fact that at some point in time, new users of a particular software program have to learn how to use the program. In fact, as Baecker, Grudin, Buxton, and Greenberg (1995) point out that, with the sophistication of computer interfaces in recent years, the need for training in the proper and efficient use of computers and software programs has become even more important.

In review, there are several important trends currently taking place regarding the use of computers in education. Hypermedia programs are becoming more common as computers with multimedia capabilities are being purchased and placed into schools. The results of a meta-analysis of 35 recent research studies indicate that the effects of using hypermedia instruction were positive when compared to traditional instruction methods. Another current trend is the movement of computers out of labs and into classrooms where they can be used by regular classroom teachers and students alike as a tool to enhance the teaching and learning of new material. The movement of computers out of labs and into regular classrooms coincides with the fact that most educators now recognize the computer as a tool to be used to enhance learning, as opposed to a subject simply to be studied. Some researchers predict that the computer will be used in the

future for more individualistic instruction. At the same time, distance education courses are becoming more prevalent, courses in which students can participate from remote locations and which can be taken and completed according to one's own timetable. Finally, no matter what software programs may be used in the future, there will still remain a need to train users in the proper and efficient use of them.

Summary of Technology Usage in the Classroom

The earliest mainframe computers were used primarily as number crunching machines to aid in solving complex calculations in the fields of mathematics, science, and engineering. The first educational use of computers began in the 1960's, when a few university classrooms made use of remote terminals connected to a mainframe computer. These computer terminals were often used for drill and practice computer based instruction (CBI) activities. It wasn't until the late 70's and early 80's that stand-alone microcomputers became available, but drill and practice techniques still dominated the available educational software.

Mathematics, a discipline where drill and practice activities were common long before the advent of the computer, seemed to be a natural place to introduce the computer into the curriculum. Various research studies produced conflicting results concerning the effectiveness of drill and practice software over traditional paper and pencil drill and practice activities. Yet, as recently as 1995, it was estimated that drill and practice software still remained the predominant form of software found in mathematics classrooms.

Many researchers and educators are now calling for the computer to be used as a tool for learning, rather than a subject to be studied. Many, such as those connected with the mathematics reform movement of the NCTM, have come to view the computer as a catalyst for changing the current view of school, the learning process, and the focus of education, from a passive, teacher-centered environment, to an active discovery oriented, learner-centered environment.

One example of how the computer can be used as a tool to enhance learning is found in the emergence of dynamic graphing software packages, such as *The Geometer's Sketchpad*. These programs are used in many mathematics classrooms and allow the user to make constructions, measure angles, and discover for themselves the essential concepts of geometry.

Other recent trends concerning the use of computers in education include programs which incorporate hypermedia and multimedia capabilities, moving computers out of the lab and into individual classrooms, and the increased popularity of distance education courses. But, no matter what the future trends in the educational use of technology may be, there still remains the fact that at some point in time, new users need to learn how to effectively use the new software programs.

Software Training

So far this chapter has examined what past research has revealed concerning cooperative learning and the use of technology in education, especially as it applies to the mathematics classroom. It has already been shown that educational researchers endorse

the use of technology and cooperative learning strategies as students learn new concepts and skills. The next part of this chapter examines recent literature and research to see what it has to say concerning the best way to instruct users to use new software programs.

This section begins by focusing on human computer interaction (HCI) as it pertains to the design, training, and use of software programs. Following this, various training issues are explored as they pertain to the learning of new software programs. The need for software training is explored, as well as current research on end-user software training methods in the business community. Next, the current state of teacher training in the use of software programs in the classroom is examined as well as the need for continued teacher training. Finally, the current state of student training methods and models are examined.

Human Computer Interaction

Before one can fully understand and appreciate the need for user training in the use of software programs, one must first take a look at how people and computers interact. This growing field of study is known as Human Computer Interaction (HCI), a field which focuses on effective ways to improve the interaction between computer users and computer systems. HCI research has led to an abundance of information concerning how computer systems can be better designed, developed, implemented, and evaluated, all in an attempt to make them more productive and easier to use by computer users. The rapid growth of computer usage in education, as well as in business and society in general, has made the study of human-computer interaction essential to good interface design (Butler, Jacob, & John, 1998; Grunwald, 1997).

Studies in HCI, as reported by Nielsen and Mack (1994), point out that the way in which a software application program is designed can greatly affect its usability. Andleigh and Thakrar (1996) point out that there is never a single perfect or "correct" interface design for a particular application, only good and bad designs. Often the perceived correctness of a particular interface design is largely dependent on the personal preferences of the individual user. For this reason, an important step in designing, and evaluating, a user interface is to know who will be using the interface (Lewis & Rieman, 1994). Likewise, a user interface must communicate clearly with the intended user if it is going to be easy to use (McFarland, 1995).

Ease of learning, or "learnability," is an important usability attribute which must be carefully considered when selecting software, especially for use in the classroom. Other attributes of usability, like functionality and ease of use, are closely correlated with ease of learning. For example, if the functionality of the program doesn't match closely to the user's needs, and the user is forced to perform arcane sequences of steps to accomplish the desired task, then the program will be difficult to learn to use (Wharton, Rieman, Lewis, & Polson, 1994). Thus, ease of learning, which includes length of time it takes to learn a software package, needs to be taking into consideration when new software packages are selected for classroom use. Just because a software package is easy to use once the basics of use are mastered does not mean that it is necessarily easy to learn (Mack & Nielsen, 1994).

Evaluating the usability of a particular software application package is often broken into two categories, formative and summative evaluations. Formative evaluations occur before the actual implementation of the new software package, often while the

software is still being developed or is in the alpha testing stage. Summative evaluations are conducted after the program has been created and is either in beta testing or has already been released to the general public. The summative stage of usability evaluation may be too late to test a program's learnability, unless the purpose of the evaluation is to compare it to other programs already on the market or to study ways in which future releases of the program can be improved. The ease of use and learnability of a software program should be tested during the formative usability evaluation process so changes can be incorporated into the program before it is released to the general public (McMillan & Schumacher, 1993; Vaughan, 1996).

Nilsen et al. (1993) conducted a longitudinal research study to help determine how software could be designed better to encourage more effective learning, and more complete and effective use of the software's powerful features. The study involved 36 entering university students, tracking their learning and use of Lotus 1-2-3 from novice towards expert use over a 16 month period. The students were taught Lotus 1-2-3 using an 11 page paper-based tutorial while seated at individual computers in a large open computer laboratory. After a brief introduction, the students worked through the tutorial individually at their own pace. The results of the study indicated that while the motor aspects of performance were relatively stable over time, the improvement in cognitive skills were dependent on aspects of menu structure and how many things had to be recalled from memory, among other things. The results implied that changing the menu structure of the program may be advantageous in order to speed the process of moving from a novice to expert user, in other words, to improve the program's ease of learning and use.

In summary, HCI involves the study of how people and computers interact. HCI research has lead to an abundance of information concerning how computer systems can better be designed, developed, evaluated, and implemented to make them easier for people to use. Research has shown that the way in which a program is designed can greatly affect its usability. A user interface must clearly communicate with the intended user if it is going to be easy to use. Ease of learning, or "learnability," is an important usability attribute which must be carefully considered when selecting software. Just because a software package is easy to use once its basics of operation are mastered, does not mean that it is easy to learn. The ease of use and learnability of a software program should be tested during the formative usability evaluation process so changes can be incorporated into the program before it is released to the general public. One research study (Nilsen et al., 1993) has demonstrated that the menu structure, among other things, can greatly affect a program's ease of learning and ease of use.

The Need for Training

Even if a program is easy to learn and use, proper training is still an important part of learning how to use the program in an effective and efficient manner. Baecker, Grudin, Buxton, and Greenberg (1995) point out that training is an essential part of any human computer interface. Without adequate training concerning the basic usage of a software program, the user will generally not make use of the program's features in the most expedient fashion.

Wharton, Rieman, Lewis, and Polson (1994) report that many users prefer to learn new software programs by exploration, as opposed to formal training sessions. But, the

research of Wiedenbeck and Zila (1997) found that open-ended exploration by the learner in learning how to use a new software package tends to be unsuccessful. So once again, the need for formal training is apparent.

However, research indicates that just because formal software training takes place is no guarantee that the software will be used once the training sessions are completed (Shayo & Olfman, 1993). One draw back to this research, however, is that much of it has been conducted in the area of software training in the business world, which is quite different from the educational arena where students often learn to use a software program for immediate application within a particular class. To add to the confusion concerning how to best train students, Schatz (1996) points out that introducing computer applications to students, while necessary, can be particularly time consuming and frustrating, and goes on to state that there seems to be no good way to introduce students to new software programs.

Although much has been written concerning the use of cooperative groups in the classroom (Saxton, 1995), and much has been written concerning effective software training techniques in the business community, there appears to be a striking lack of research which looks specifically at the use of cooperative groups associated with the learning of new software programs in the educational arena, a research deficiency also report by Bailey (1997).

In review, researchers point out that training is an essential part of any human computer interface, and that training is necessary if end-users are going to learn and use the program's features in the most expedient way. Users, however, prefer to learn new software through open exploration, which researchers have shown to be an inefficient

method of learning. Also, if formal training sessions do take place, there is no guarantee that the training will carry over into actual use. Some educators have even lamented that while software training is necessary, there seems to be no good way to introduce students to new software programs, while some researchers have commented that there appears to be a striking lack of research which looks at the utilization of cooperative groups during the training process, either in the business or educational communities.

Training Techniques in the Business Community

As previously noted, much has been written concerning software training techniques used in the business community. Because some of this research may have implications for software training in the educational community, this section examines some of these current training techniques.

Gist, Schwoerer, and Rosen (1989) compared two alternative computer software training methods to discover what effect they might have on self-efficacy and performance of the user. Their field experiment involved 108 university managers learning how to use a popular spreadsheet program. The two training methods compared in the study were a behavioral modeling approach and a tutorial approach.

The subjects involved in the behavioral modeling approach watched short video taped segments in which a person modeled the specific steps needed to perform specific program functions. Following each short video clip, the subjects performed the functions individually on separate computers. The subjects involved in the tutorial training approach worked individually using a one-on-one interactive tutorial diskette. The diskette told the participants what to do, but did not model any of the steps for them.

The results of the study indicated that the participants involved in the behavioral modeling approach outperformed the participants in the tutorial training approach on an objective measure of computer software mastery. Also, the behavioral modeling approach participants reported a higher degree of satisfaction with the training process, a more positive work style, and less negative effect during training than did the tutorial trainees at all levels of computer self-efficacy.

Even though much research has been conducted concerning improving the efficiency of software training in the business world, there is still the problem of the trainees bringing their learning back to the work place and applying it to their jobs. Two separate training studies, Olfman and Bostrom (1991) and Olfman and Mandviwalla (1992), report that less than half of the workers who attend formal software training sessions actually end up using the software later on.

Shayo and Olfman (1993) conducted qualitative methods to try to discover why end-users of formal software training sessions did not apply their learning to their jobs once they returned to the day to day activities of the work place. They discovered that there was often a lack of alignment between the trainees' pre-training goals and the trainer's goals. Shayo and Olfman suggest that trainers need to know the needs of the trainees in advance, use job related examples during the training process, utilize hands-on training methods, and provide after training support. They also note that it may not be possible to structure formal group training to meet the individual goals of trainees because of the varying backgrounds and job positions of the trainees, in which case, one-on-one training to address specific job needs may be more beneficial.

In a subsequent paper, Shayo and Olfman (1994) reiterate their conclusion that the lack of transfer of software training from formal end-user training sessions to the work place usually does not result from individual trainee characteristics, features of the software program, poor training methods, nor the social context of work. Rather, poor transfer occurs because of a lack of goal matching between the trainee and the training program.

Although goal matching is important in the business community, it is not as great of a factor in the educational community because the students' reasons for learning a particular software program and the teacher's reason for teaching it should be the same, to use the software application as a tool for other work in the current class or future classes. Still, introducing a new software program to students can be very time consuming and frustrating, both for the teacher and the students (Schatz, 1996).

In summary, one research study in the business community has shown that a modeling approach to software training where the use of the software is demonstrated to the participants as they learn how to use it, as opposed to a tutorial approach where no modeling is performed, yielded higher performance results of computer software mastery. The modeling approach participants also reported a higher satisfaction level with the entire training process. Several other studies found that a lack of transfer from the training session to the actual jobs of the participants often occurred because of a misalignment between the goals of the trainees and trainer. It was also noted that one-on-one individualized training methods may be more advantageous in the business community because of the varied background and future goals of the trainees. Goal matching, although a huge concern in the business community, is not seen as a major

concern in the educational community because students are often learning to use a software package for an immediate application in a particular class.

Current State of Teacher Training

We now turn our attention to the state of teacher training as it pertains to the use of technology in the classroom. If teachers are not trained in the proper use of technology in the classroom, then training students to use technology appropriately becomes an even greater task. This is the case because the students wouldn't have had the prior experience of observing teachers correctly using and modeling the technology in the classroom.

While technology training budgets for businesses increased nearly 20 percent in 1998, trainers and training materials were considered to be at a critical shortage (Appleton, 1998). The situation for educators is even more bleak. In 1996, more than 50 percent of recent graduates from teacher-education schools reported that they were either not prepared or poorly prepared to use information technology in the classroom. Only three percent of the graduates described themselves as "very well prepared" (Barksdale, 1996). This means that not only are school districts required to train their current staff in the appropriate use of technology in the classroom, but many of the recently graduated teachers as well.

Many school districts are now learning first hand that teachers need training to successfully integrate technology into their classrooms (Holzberg, 1998). It is a known fact that teachers need training because they tend to teach the way they were taught (Rasmussen, 1996), which did not include the use of technology. Many of them have

never experienced a learning environment where technology was used as a tool to enhance learning.

The most common training method used among educators is the faculty inservice or one-day workshop. But, research has shown that one-shot training sessions have little long-term effect on classroom practice, and are therefore not adequate in and of themselves (Wiburg, 1994). The same can be said for on-line tutorials because teachers need continuous support to apply and retain what they have learned about the use of technology in the classroom (White, 1995). Tally and Grimaldi (1995) and Lovely (1997a) report that the support and encouragement of fellow staff members is crucial to the successful implementation of any new technology usage. Lovely suggests several strategies to keep the momentum going between staff development sessions, such as ongoing mini-courses, participant newsletters, e-mail networks, student support cadres, accessability to printed materials, and face-to-face visits.

However, continued support by itself may not be sufficient. The training in technology usage must also be practical and pertinent to the current needs of the trainee. This has long been a common criteria for industry software trainers (Boyd, 1998), and obviously applies to the educational arena as well. Teachers, and students, need to see a direct connection between what is being taught and how they can use the new technology. In other words, the application they are learning must be relevant to their current needs (Lovely, 1997b). Atkins and Vasu (1998) point out that far too often faculty training sessions consist of a workshop on how to use some new software program without any common explanation concerning how the program can or should be used to help meet the current curricular objectives.

Some technology coordinators feel that the best way to get teachers to use technology with their students is to have the teachers first learn to use the computer for personal applications. Wetzel (1996-97) points out that this model of initial technology training, which has been used for nearly a decade, is a fallacy. Wetzel maintains that if you really want teachers to use the technology with their students, then you need to find curricular-based software which can be modeled for them in teacher inservices.

Not only does technology training need to be relevant to classroom experience, but it is also beneficial if teachers can actually practice the new technology application during the training session. In other words, teachers will more easily learn to apply new techniques and applications to their own classrooms, if they can learn about them in a hands-on environment (Harrington-Lueker, 1996).

Even though the use of small cooperative groups is widely accepted by educators as a valid means to increase learning of new materials, very few training session make use of cooperative groups (Siegel, 1995). In a 1995 survey of teachers involved with training sessions, Siegel reports that only 32 percent of the respondents reported the use of small groups with hands-on learning. Rasmussen (1996), also points out that working in small groups when learning new applications of technology is often more effective than working alone. In a review of the available literature, only one report of a workshop where participants worked in pairs was located. Emmans and Byxbee (1997) reported on a workshop of Brazilian teachers where many of the participants found working in pairs to be the ideal arrangement.

In summary, the current state of teacher training in the use of technology in the classroom is very inadequate. Teachers do not learn to apply technology by simply

reading or listening to lectures about it, rather, teachers need to see the technology application modeled, and then need to be trained to use the technology effectively in their classrooms. Research has found that training is most effective if it is hands-on and done in small groups. Also, technology training for teachers is most effective if it is relevant to the current needs of the teacher and can be immediately applied in the classroom. Technology training must be followed up with additional support, as opposed to a one time shot, if it is to have a lasting effect on teacher usage in the classroom.

Current State of Student Training

While there is adequate research on training workers in the business world to use new software packages, and some research on training teachers in the use of software applications for the classroom, the research literature concerning the most efficient way to introduce students to new software applications is very sparse. This section examines what has been found concerning the training of students to use new software packages.

Bailey (1997) agrees that while there seems to be an abundance of research related to computer learning, there is little information available concerning student learning with emerging technologies, such as electronic cooperative learning. Sullivan (1994), while discussing computer technology and collaborative learning as it applies to teaching writing in electronic classrooms, simply mentions that it is up to the teacher to decide how much class time can be devoted to the learning of new software, without mentioning any particular method to use in the training process.

Several studies have been completed which do take a look at comparing various software training methods for use in the classroom. Bohlen and Ferratt (1993) compared

the effect of learning style and method of instruction on student achievement, efficiency, and satisfaction of end-users learning a new computer software program. The subjects in the study, consisting of 120 business students in an introductory computer course, were trained in the use of WordPerfect 5.1.

The participants were trained in the use of the software either through the typical lecture method or a computer-based training (CBT) method. Students in the lecture group, over the course of seven class sessions, met in a large auditorium and listened to the instructor explain and demonstrate how to use the program WordPerfect 5.1, making use of the chalkboard, overhead transparencies and a computer projection device. Students in the computer-based training session meet in a computer lab and individually worked through a CBT package for the seven class sessions. All students in both groups were required to complete several assignments, a multiple choice test on paper, and complete a computer based test to demonstrate their ability to use WordPerfect 5.1.

The results of the study indicated that those students who participated in the CBT method of instruction obtained a higher level of achievement, efficiency, and satisfaction in learning how to use the new software program than those students who participated in the lecture method of instruction. The same results were evident when preferred learning styles of the students were taken into consideration, except for students whose learning style was classified as assimilator. Assimilators appeared to learn equally well under the lecture method and the CBT method of instruction.

Chia and Duthie (1993) conducted a study with primary school children and computer-based art work. Part of the focus of their study was to evaluate how well students learned in a cooperative work environment, but an evaluation of how the

students went about learning to use the particular software package was not a part of the study. Twenty 11-year-old students participated in the study and were selected on their high grades in art. The study consisted of eight three-hour sessions, spread out over an eight week period, in which the students used a computer drawing program to experiment with creating art work.

What is interesting about this study is that it's one of only a few studies which used cooperative pairs to group the students while working at a computer. During the first two week period, it was observed that in seven of the ten pairs, one of the students in the pair would tend to dominate the use of the mouse and in some cases refused to share it with the other student. However, when a separate computer was given to the students who were not being allowed to use the mouse, those students also made rapid progress working alone. The researchers also observed that combined gender groups and paired girls-only groups worked better than paired boys-only groups. The student pairs were encouraged to, and were often observed, discussing and critiquing their work in progress.

Not all studies have found the use of cooperative learning strategies while learning to use a new software program to be advantageous. Chiu (1995), while conducting a study comparing cooperative learning verses whole-group instruction with Asian undergraduate students, concluded that the cooperative learning method itself possesses no distinct advantage over the individual learning method for office automation software training. However, it needs to be noted that the cooperative learning teams consisted of three to four students each, with the instructor serving as the facilitator. No mention is made by the researcher indicating if all team members shared the same computer or if each had their own to work with. Also, the whole-group instruction

appears to have been conducted in a large lab type setting, but, with each student working individually at a computer. Therefore, it is hard to make a comparison between this research and other studies which have been done.

Wiedenback and Zila (1997) conducted extensive research focussing on whether learners are able to use exploration-based practice methods effectively to learn to use new software and whether some minimal computing background is necessary to be successful with minimalist training and exploration practice. The empirical study compared exercise, exploration, and a combination of the two methods to see which would work best in learning to use new software, both for the novice computer user and the experienced user. The participants in the study consisted of 102 university students, divided into six groups based upon experience level and method of training. The software program used in the study was HyperCard.

The training sessions consisted of a single 2 ½ hour session, with approximately 90 minutes for training and 60 minutes for evaluation. All of the participants in each of the groups worked individually through the same instructional exercises, but then were given different instructions during the practice phase at the end of each exercise. Some students were given explicit exercises to perform for practice, some were just instructed to explore the new features of the program on their own, and some were given a combined format.

The results of the study showed that for low-experience users, the type of practice made no significant difference. However, high-experience users who were trained using exercise practice or a combined form of practice did significantly better than those users who were trained using exploration practice only. The researchers suggest that the reason

that free exploration practice of the new concepts learned did not produce the same positive results as exercise practice is that under free exploration, students tend to go off in various directions and don't focus closely on the task at hand.

While specific research which directly examines the use of cooperative pairs while training students to use a new software program appears to be very sparse, a review of the currently available literature has uncovered one teacher's model for training students in the use of new software tools. Schatz (1996), a professor of Instructional Technologies at San Francisco State University, has devised a method for training his students in the use of new software tools, a method he calls Show/Do/Cue.

Five basic steps comprise Schatz's model. 1) Provide a very brief introduction to the class concerning the new software they are about to learn. 2) Show the class a finished project created with the software. 3) Quickly, five minutes or less, build a chunk of the finished product while the students observe and get an overall picture of the procedure. 4) Handout to the students a detailed set of instructions on how to complete the entire project. The students then work in cooperative-pairs to create their own project during class. 5) Finally, give the students a computer-based tutorial that guides the students through the same procedure so the students can practice on their own outside of class.

In summary, the available research appears to be very sparse concerning training students in the use of new software programs. One study did compare the use of computer-based-training (CBT) methods to traditional lecture methods when training students to use a new software program. The participants in the CBT method of instruction obtained a higher level of achievement, efficiency, and satisfaction than did

those participating in the lecture method. Another study compared the use of an exercise format to a free-exploration type format during the practice phase of learning to use a new software program. The results indicated that the training format made no difference for novice users, but experienced users did better when they used the exercise format. It is interesting to note that both of these studies, as did most, made use of students working as individuals, rather than student pairs, as they learned to use the new software.

A review of the current literature discovered one teacher's model for training students in the use of new software packages, the Show/Do/Cue method of Schatz (1996). However, no specific research which directly examines the use of cooperative pairs while training students to use a new software program was located.

Summary of Software Training

HCI research has pointed out that the design of a program can greatly influence its learnability and usability. However, the learnability of a program should be tested during its formative stages, rather than after it has been created and released for distribution to schools. Researchers have also pointed out that training is an essential part of any human computer interface, and that training is necessary if users are going to learn to use the software features in an expedient fashion.

Some educators have expressed the thought that there is no good way to train students to use a new software package. Other researchers have commented that there seems to be very little research that has been done in the area of training students to use new software packages. Although much of the research conducted within the business community may not be directly transferable to the educational arena, some general

lessons can be learned from this research. One such lesson is that a modeling approach, with hands-on participation by trainees, produces better results than tutorials alone.

Before teachers can begin to train students, they too need to be trained in the effective use of the software program. Research has shown that teachers learn best if the training is hands-on, done in small groups, is relevant to their current needs, and can be immediately applied in the classroom. Follow up support is also extremely important in order to achieve a lasting effect on teacher usage.

Although much has been written concerning the benefits of using technology in the classroom, and even a fair amount of material can be found on training teachers to use software in the classroom, the literature dealing specifically with training students to use new software programs effectively appears to be very sparse.

Summary of Literature Review

This section summarizes what was previously stated concerning what is known and unknown about the use of cooperative learning techniques during the process of training students to use new software programs. The section is divided into four parts, the first three summarizing what the currently available literature and research has to say about cooperative learning, technology in education, and the training of users to use new software programs. The final part reiterates what the review of literature has revealed and has not revealed.

Cooperative Learning Usage in Education

Cooperative learning was defined by Johnson and Johnson (1989) as the instructional use of small groups where students work together to maximize their own and each other's learning. Serra (1997) provided a similar definition stating that cooperative learning, or cooperative small group instruction, referred to classroom techniques in which students worked together in small groups on learning activities and received recognition based on the group's performance.

Cooperative learning techniques are not something new to the educational community, but interest in and the use of cooperative groups has increased in recent years, partially due to the use of technology (Roschelle, 1994; Strommen, 1995). Many researchers have found that the use of cooperative groups consistently yields superior results when compared to other learning techniques, and leads to the development of leadership skills, improved social skills, a sense of teamwork, and improved self-esteem, as reported by Strommen and others.

Johnson and Johnson (1989), have identified five essential elements of effective cooperative learning structures. First, positive interdependence is essential. The cooperative learning experience must be designed so that all participants contribute to the collaborative task of the group and each member of the group feels needed. Secondly, face-to-face promotive interaction needs to be practiced. Group members, in face-to-face gatherings, promote each other's learning by helping, encouraging, and supporting one another during the cooperative learning experience. Third, individual accountability is required. Students, while working together to complete the cooperative task, still need to be held accountable for, and assessed on, their own individual learning, as well as accountable for their role in the group effort. Fourth, students need to develop

interpersonal and small group skills in order to be effective members of a cooperative group. Fifth, group processing time, time for the students to evaluate how well they are working together as a group, must be provided.

Cooperative groups can occur in a variety of formats in the classroom. Research has found that cooperative groups of four or five members work best for most cooperative learning situations (Serra, 1997), but cooperative pairs seem to work best when working on computer activities (O'Malley, 1992). Groups can be created in various ways, but research has found that student selected groups are usually the least productive (Jewett, 1996; Johnson, Johnson, & Smith, 1991) and teacher assigned heterogeneous groups, based on student ability, are the most productive and beneficial to all participants (Serra, 1997). Groups should be changed occasionally so students learn to work with other groups too (Foster, 1993), and each student within a group should have a specific role assigned to them in order to create a sense of responsibility to the group (Johnson & Johnson, 1989).

Cooperative learning has become popular in the mathematics classroom in recent years. The National Council of Teachers of Mathematics (NCTM) is seen as a major force behind the mathematics reform movement, which embraces the use of cooperative learning techniques for the mathematics classroom (NCTM, 1989, 1991). Some educators go even so far as to say that cooperative learning techniques should be the main stay for all mathematics instruction, especially those who are strong supporters of the constructivist view of mathematics learning (Dubinsky & Schwingendorf, 1997). Others caution that cooperative learning is only another technique in the arsenal of tools which mathematics teachers can make use of in the classroom (O'Connor, 1997).

The use of cooperative learning techniques has also coincided with an increased use of technology in the classroom. Many educators and researchers now view the computer as a tool which will enhance the use of cooperative learning in the classroom (Strommen, 1995; Tomlinson & Henderson, 1995). In fact, some educators now take it for granted that students should be using computers and working in groups as a normal part of the learning environment (McMahon, 1990). The use of technology in group work can be broken into three levels of use; "learning around the computer", "learning with the computer", and "learning mediated via the computer" (O'Malley, 1992). Researchers have also discovered that the use of computers along with cooperative learning strategies is an excellent way for students to develop problem solving skills (Denning & Smith, 1995; Natasi, Battista, & Clements, 1990).

The use of technology and cooperative learning has been particularly prevalent in mathematics classrooms in recent years. Many educators, researchers, and authors now point to the use of technology as a tool to be used in conjunction with cooperative learning strategies to support the teaching and learning of mathematics (Ulslick & Walker, 1994). As more mathematics teachers embrace discovery learning, as associated with constructionist theories of learning, they are encouraging their students to work in cooperative groups where computers are being used as a tool of discovery.

Technology Usage in the Classroom

The use of computers in education has had a relatively brief history. The first operational mainframe computers were used primarily as number crunching machines to aid in solving complex calculations in the fields of mathematics, science, and engineering

(Withrow, 1997). The first educational computer revolution occurred in the 1960's when time-sharing systems, with remote terminals for student use, were developed and located on a few university campuses (Dockterman, 1997). During this same time period, the Beginners All-Purpose Symbolic Instruction Code (BASIC) programming language was developed and was often used to develop drill and practice, computer based instruction (CBI) materials. These early CBI materials enabled students to progress at their own pace along individualized paths of learning, but were not widely used (Molnar, 1997). A second computer revolution began in the late 1970's or early 1980's when the stand-alone microcomputer became available to schools. However, drill and practice techniques still dominated the available educational software.

The early use of computers in education was most heavily concentrated in the area of mathematics education. The software programs used were primarily of the drill and practice variety because drill and practice was a common method of review used in the mathematics classroom, even before the advent of computers. Various research studies have resulted in conflicting findings concerning the effectiveness of drill and practice software over traditional paper and pencil drill and practice activities (Casey, 1987). Yet, as recently as 1995, it was estimated that drill and practice software still remained the predominant form of software found in mathematics classrooms (Strommen, 1995).

Many educators feel that the computer should be used for more than just running drill and practice or tutorial software, or simply as an object to be studied in a computer science class (Salisbury, 1985). Many reformers, such as Perelman (1992), Papert (1993), and others, see the computer as a catalyst for changing the current view of school, the learning process, and the focus of education, from a passive, teacher-centered

environment, to an active, discovery oriented, learner-centered environment. These educators are calling for the computer to be viewed and used as a tool in all subject areas, by teachers and students alike, to enhance the learning process. These same educators are asking that computers be moved out of computer labs and into the classroom where they can be used as a tool by all (Griest, 1996; Kennedy, 1996). But even if this happens, the question still remains of how to best train students in the use of new software programs.

The recent mathematics reform movement has caused many mathematics teachers to move towards a view of mathematics instruction which is student-centered, where the student is an active participant involved in social interaction, constructing, discovering, conjecturing, and analyzing mathematical concepts (DeCorte, 1992). The NCTM (1989, 1991) calls for technology as one of the tools which teachers should use to help meet this vision of a student-centered constructivist classroom.

Even though some research studies, which compared computer assisted instruction (CAI) to more traditional methods of instruction in the classroom, did not find CAI to lead to greater learning, they did find that CAI methods did lead to a more positive learning experience for the students (Alexander, 1991; Barnes, 1991). In addition, one study found that even though students preferred the teacher's explanation of geometry concepts over an artificial CAI geometry tutor's, the students still preferred to work with the artificial tutor because of the anonymity involved. The students also appreciated the change in the teacher's role from the main provider of information to an individual resource person (Britt, Eurich-Fulcer, & Schofield, 1994).

The focus of learning in the mathematics classroom has also shifted away from the memorization of rules to one of problem solving skills (NCTM, 1989, 1991). To aid

in this effort, research has shown that technology, along with the appropriate software, can be used to help facilitate problem-solving by students (Enderson, 1997; Widmer & Sheffield, 1998).

The use of dynamic graphing software is one example of how the computer can be used as a tool to enhance the learning of a particular subject. Graphing software was first introduced in the early 1980's in the form of the programming language Logo, a program which was seldom used as first envisioned by its author (Ploeger, 1984). The current generation of geometrical drawing software emerged in the early 1990's, and can be used as a tool by students as they explore various concepts connected with the study of geometry. Through the use of these software programs, students are encouraged to explore, construct, investigate, conjecture, analyze, and rediscover various geometric concepts (Cuoco, Goldenberg, & Mark, 1994). These new software programs have been rapidly adopted for use in high schools and middle schools even though little research, if any, has been conducted concerning the effect that these geometrical drawing programs have on student achievement. Because teachers tend to teach in the same manner that they were taught (Rasmussen, 1996), it is important for teachers to experience first-hand these new tools of learning before they employ them in the classroom (Holzberg, 1998).

There are several important trends currently taking place regarding the use of computers in education. Hypermedia programs are becoming more common as computers with multimedia capabilities are being purchased for schools. The results of a meta-analysis of 35 recent research studies indicate that the effects of using hypermedia instruction were positive when compared to traditional instruction methods (Liao, 1998). Another current trend is the movement of computers out of labs and into classrooms

where they can be used by regular classroom teachers and students alike as a tool to enhance the teaching and learning of new material. This relocation of computers coincides with the fact that most educators now recognize the computer as a tool to be used to enhance learning, as opposed to a subject simply to be studied (November, 1997). Some researchers predict that the computer will be used in the future for more individualistic instruction and instruction on demand (Bork, 1997; Hawkins, 1997; Smith & Tarkow, 1998). Distance education courses are also becoming more prevalent, courses which students can participate in from remote locations and which can be taken and completed according to their own timetable (Kelly & Leckbee, 1998; Withrow, 1997).

No matter what the future trends in the educational use of technology may be, there still remains the fact that at some point in time, new users of a particular software program have to learn how to use the program. In fact, as Baecker, Grudin, Buxton, and Greenberg (1995) point out that, with the increase in sophistication of computer interfaces in recent years, the need for training in the proper and efficient use of computers and software programs has become even more important.

Software Training

Human computer interaction (HCI) research has lead to an abundance of information concerning how computer systems can better be designed, developed, evaluated, and implemented to make them easier for people to use (Butler, Jacob, & John, 1998; Grunwald, 1997). Research has shown that the way in which a program is designed can greatly affect its usability (Nielsen & Mack, 1994). A user interface must clearly communicate with the intended user if it is going to be easy to use (McFarland,

1995). Ease of learning, or "learnability," is an important usability attribute which must be carefully considered when selecting software (Wharton, Rieman, Lewis, & Polson, 1994). Just because a software package is easy to use once its basics of operation are mastered, does not mean that it is easy to learn (Mack & Nielsen, 1994). One research study, by Nilsen et al. (1993), has demonstrated that the menu structure, among other things, can greatly affect a program's ease of learning and ease of use.

Researchers point out that training is an essential part of any human computer interface, and that training is necessary if end-users are going to learn and use the program's features in the most expedient way (Baecker, Grudin, Buxton, & Greenberg, 1995). Users, however, prefer to learn new software through open exploration (Wharton, Rieman, Lewis, & Polson, 1994), which researchers have shown to be an inefficient method of learning (Wiedenbeck & Zila, 1997). Also, if formal training sessions do take place, there is no guarantee that the training will carry over into actual use (Shayo & Olfman, 1993). One major drawback concerning software training research is that much of it has been conducted in the business community, and therefore might not be transferable to the educational community.

Some educators have even lamented that while software training is necessary, there seems to be no good way to introduce students to new software programs (Schatz, 1996). Other researchers have commented that, although much has been written concerning the use of cooperative groups in education (Saxton, 1995) and software training in the business community, there appears to be a striking lack of research which looks at the utilization of cooperative groups during the training process, either in the business or educational communities (Bailey, 1997).

Even though research studies conducted in the business community may not be directly transferable to the educational community, never the less, some lessons learned in the business community concerning end-user training are worth noting. One research study in the business community, conducted by Gist, Schwoerer, and Rosen (1989), has shown that a modeling approach to software training where the use of the software is demonstrated to the participants as they learn how to use it, as opposed to a tutorial approach where no modeling is performed, yields higher performance results of computer software mastery. The modeling approach participants also reported a higher satisfaction level with the entire training process. Several other studies found that a lack of transfer from the training session to the actual jobs of the participants often occurred because of a misalignment between the goals of the trainees and trainer (Olfman & Bostrom, 1991; Olfman & Mandviwalla, 1992). Also, one-on-one individualized training methods may be more advantages in the business community because of the varied background and future goals of the trainees (Shayo & Olfman, 1993). Goal matching, although a huge concern in the business community, is not seen as a major concern in the educational community because students are often learning to use a software package for an immediate application in a particular class.

In order for teachers to properly use and model technology in the classroom, they first need to be trained in its use. However, the current state of teacher training in the use of technology is very inadequate (Barksdale, 1996). Teachers do not learn to apply technology by simply reading or listening to lectures about it, rather, teachers need to see the technology application modeled and then need to be trained to use technology effectively in their classrooms (Holzberg, 1998; Rasmussen, 1996). Research has found

that training is most effective if it is hands-on and done in small groups, is relevant to the current needs of the teacher, and can be immediately applied in the classroom. Also, technology training must be followed up with additional support, as opposed to a one time session, if it is to have a lasting effect on teacher usage in the classroom (Atkins & Vasu, 1998; Boyd, 1998; Harrington-Lueker, 1996; Lovely, 1997a, 1997b; Tally & Grimaldi, 1995; White, 1995; Wiburg, 1994).

Although, very little research has been done concerning training students in the use of new software programs (Bailey, 1997), one study, carried out by Bohlen and Ferratt (1993), did compare the use of computer-based training (CBT) methods to traditional lecture methods when training students to use a new software program. The participants in the CBT method of instruction obtained a higher level of achievement, efficiency, and satisfaction than did those participating in the lecture method. Another study, conducted by Wiedenback & Zila (1997), compared the use of an exercise format to a free-exploration type format during the practice phase of learning to use a new software program. The results indicated that the training format made no difference for novice users, but experienced users did better when they used the exercise format. It is interesting to note that both of these studies, as did most, made use of students working as individuals, rather than student pairs, as they learned to use the new software.

During the review of literature, only one teacher's model for training students in the use of new software packages was discovered, the Show/Do/Cue method of Steve Schatz (1996). However, specific research, which directly examines the use of cooperative pairs while training students to use a new software programs, appears to be very limited.

What is Known and Unknown

As stated in the previous sections, the currently available literature, as supported through recent research, speaks volumes concerning the benefits of using cooperative learning in the classroom, but is almost silent concerning the appropriate use of cooperative learning during training sessions for new software users. Likewise, the current literature and research is quite vocal concerning the benefits of using technology in the classroom, and even speaks at length on how teachers can use software tools in the classroom to enhance learning, but is strikingly less vocal when it comes to the question of how to get students started with these new software programs. Finally, the current literature and research has much to say concerning training end-users in the business community, and speaks forcefully about the need to train teachers, and even offers guidelines for training teachers to use new software programs in the classroom to enhance learning, but is almost silent concerning effective and efficient methods for training students to use new software programs.

The current literature indicates that there is a need for more effective and efficient methods to train students to use new software programs. However, current research has not yet fully responded to that need. Research, which directly examines the use of cooperative pairs while training students to use a new software program, appears to be very sparse. It is not currently known if the use of cooperative groups, while learning to use new software, will decrease the amount of class time needed to learn the basics of the software program, and allow more students to master the basics of using the software effectively.

Contribution This Study Will Make to the Field of Software Training

The previously stated purpose, or goal, of this study was to increase understanding of how the use of cooperative groups affects both the mastery of the basics of a software program and the amount of time needed by students to learn the basics of a software program, so the program can then be used to enhance learning of the subject matter. The specific research questions are:

1. Do students who cooperatively learn to use a software program, learn to use the program significantly faster than students who independently learn to use the same software program?
2. Are students who cooperatively learn to use a software program significantly more likely to master the basics of the program's use than students who independently learn to use the same software program?

An extensive review of literature on the subjects of the use of cooperative learning techniques in the classroom, the use of technology in education, and current training methods associated with training in the use of new software programs has been conducted. This review of literature has failed to answer the above questions, thus the original problem of how to best train students in the use of a new software program remains unanswered.

Without this study, it would still be unknown if the use of cooperative groups, while learning to use new software, would decrease the amount of class time needed to learn the basics of the software program, and would allow more students to master the basics of using the software effectively. If the amount of time needed to learn to use the

software could be reduced through the use of cooperative groups, then more time would be available to use the software as a tool to enhance learning during class. Also, if the percentage of students who master the basics of using the software increases through the use of cooperative groups, then less time would be needed for relearning the software at a later date.

If it can be shown, through research, that the use of cooperative groups does speed up the learning process of new software and/or increases the retention of the basics needed to use the software program effectively, then schools may need to take a fresh look at how they train their students to use new software packages. Such a finding could have enormous ramifications on the training methods employed by teachers throughout the educational community.

Chapter III

Methodology

The purpose of this chapter is to delineate the ways in which the investigation was conducted. Specifically, the hypothesis to be tested is presented, followed by background information concerning the selection and composition of the sample population which was used in the study. The instructional and test instruments are discussed, along with their reliability and validity, as well as the step by step procedure which was followed during the research study. The chapter concludes with a discussion of the projected outcomes of the study and the generalizability of the results to other populations.

Hypotheses to be Tested

As stated previously, the purpose, or goal, of this study was to increase understanding of how the use of cooperative groups, while learning to use a new software program, affects both the mastery of the basics of the program as well as the amount of time needed by students to learn those basics. The specific research questions were:

1. Do students who cooperatively learn to use a software program, learn to use the program significantly faster than students who independently learn to use the same software program?

2. Are students who cooperatively learn to use a software program significantly more likely to master the basics of the program's use than students who independently learn to use the same software program?

Based on a through review of the currently available literature, as presented in the previous chapter, it was shown that there is a need for more effective and efficient methods to train students to use new software programs. However, research which directly examines the use of cooperative pairs while training students to use a new software program appeared to be very sparse. Prior to this study, it was not known if the use of cooperative groups, while learning to use new software, would decrease the amount of class time needed to learn the basics of the software program, and allow more students to master the basics of using the software effectively.

Based on a review of the current literature and research, as it pertained to the research questions, the following hypotheses are presented:

Hypothesis 1: Students who cooperatively learn to use a new software program, will learn to use the program significantly faster than students who independently learn to use the same program.

Hypothesis 2: Students who cooperatively learn to use a new software program, will be significantly more likely to master the basics of the program's use than students who independently learn to use the same program.

Background for the Study

This section of the chapter takes a closer look at the demographics of the school system and the school where the research study was carried out. Background information is also provided as it pertains to the selection of the sample population for the study, as well as a look at the facilities which were used during the study.

Demographics of the School System

The study was conducted at Michigan Lutheran Seminary, which was part of the Wisconsin Evangelical Lutheran Synod's (WELS) ministerial education system of schools. This system of schools consisted of two preparatory schools (grades 9-12), Michigan Lutheran Seminary, located in Saginaw, Michigan, and Luther Preparatory School, located in Watertown, Wisconsin; one college of ministerial education, Martin Luther College, located in New Ulm, Minnesota; and one seminary, Wisconsin Lutheran Seminary, located in Mequon, Wisconsin. This system of ministerial schools received no public funds for its operations, but was supported solely by the over 1200 congregations of the WELS. The main purpose of this system of ministerial education schools was to provide future pastors, teachers, and staff ministers for the churches, missions, and other elementary and high schools of the WELS (P. T. Prange, Personal communication, May 22, 1998).

Demographics of the School

The study was conducted at Michigan Lutheran Seminary, a parochial high school, grades 9 through 12, located in Saginaw, Michigan. At the time of the study the school had a student population of 342, which was comprised of 98% non-Hispanic White, with the remaining 2% from a variety of foreign countries and ethnic backgrounds. The school limited entrance based on various criteria, one of which was a satisfactory score on a mathematics and English screening test. The school further limited entrance to those students who stated a willingness to consider careers as pastors or teachers in the Wisconsin Evangelical Lutheran Synod's churches or schools. Typically, between 40 and 50% of the high school's graduates continued their studies at the synod's college of ministry, Martin Luther College, in New Ulm, Minnesota. Nearly 100% of the remaining graduates entered other secular colleges and universities.

The high school also provided dormitories for approximately two-thirds of its students, with the remaining students commuting daily from home. The dormitory students had mandatory supervised study periods each evening during which time they had access to the school's networked computer lab. Commuting students could also use the computer facilities each evening.

The school had a very structured and narrowly focussed curriculum which allowed the students few electives. The typical student took four years of religion, English, mathematics, social studies, and music; three years of science and a foreign language; two years of a second foreign language and physical education. Even though the curriculum was rather limited, the mathematics curriculum consisted of the traditional four year

sequence of algebra I, geometry, algebra II - trigonometry, and pre-calculus (P. T. Prange, Personal communication, May 22, 1998).

Selection of the Population

The 10th-grade population of the school was selected to be subjects for the study because the 10th-grade was the grade level at which the software package used in the study, *The Geometer's Sketchpad,* was first learned and then used in class as a tool by the students.

Students included in the study were those enrolled in geometry at Michigan Lutheran Seminary at the beginning of the school year. Excluded from this group of participants were those students whose parents withheld parental permission for participation, students who were retaking geometry because they failed it the previous year, students with prior experience in the use of *The Geometer's Sketchpad* program, students in the English as a Second Language (ESL) program, and foreign exchange students where English is not their national language.

The school randomly scheduled its students into various sections of geometry. The study was therefore organized with one control group and two experimental groups. The same teacher taught the use of the software package, *The Geometer's Sketchpad*, to all three sections, with each section meeting in the same classroom. The three sections of geometry, each with between 24 and 26 students, was randomly selected by the researcher to serve as either control or experimental groups.

Facilities

The research was conducted in Michigan Lutheran Seminary's main computer lab, which had 30 identical Window's based 586 computers running on a single network. All of the students participating in the research were familiar with the room and the use of the computers because all of them had taken a one semester keyboarding course taught in the same computer lab the previous year. Many of the students also used the computer lab on a regular basis for other course work.

Instrumentation to be Used

The software package used in the research was *The Geometer's Sketchpad* by Jackiw (1990) and published by Key Curriculum Press, Berkeley, California. In order to be trained in the use of the software program, the students in the study used printed tutorial materials available through the publisher of the software program. The actual tutorial materials used were "Guided Tour I (Windows): The Freehand Tools" and "Guided Tour II (Windows): The Centroid of a Triangle" from Bennett's book *Exploring Geometry with the Geometer's Sketchpad* (1996), published by Key Curriculum Press. These printed materials consisted of three to four pages of typed instructions in a step by step format. Because these instructional training materials were available through the publisher of the software program, and developed to be used in conjunction with learning how to use the program, the materials had been field tested and had been found to be reliable and valid by the publisher.

The researcher created a post test instrument which was used to measure mastery of the basic components of the software package as presented during the two days of training (see appendix A). The post test instrument consisted of two activities, one corresponding to each of the two days of training, which required the participants to demonstrate their ability to use the software program. The reliability and validity of the post test instrument were verified by the author of the tutorial materials, Dan Bennett, who was also a representative of the publisher of the software package, Key Curriculum Press (see appendix B).

Specific Procedures to be Employed

The research design was quasi-experimental in nature due to the fact that the treatment was randomly assigned to pre-existing groups. The following sections delineate the specific procedures which were followed to conduct this research. These steps will allow others to replicate this research study, if so warranted.

Pre-Test Procedures

In order to conduct a controlled research study, the following steps were carried out prior to the actual treatment of the experimental groups and subsequent assessment of the learning that took place.

1. No pre-test was administered because the students in the study consisted of an entire class of geometry students, which was representative of a typical class of students who had not used *The Geometer's Sketchpad* program before.

2. The students excluded for participating in the study were identified as those whose parents withheld parental permission for participation, students retaking geometry because they had failed it the previous year, students with prior experience in the use of *The Geometer's Sketchpad* program, foreign exchange students where English was not their national language, and students who had or were participating in the English as a Second Language (ESL) program at the school.

3. Because the school randomly scheduled its students into various sections of geometry, the study was organized with one control group and two experimental groups. The three sections of geometry, each with between 24 and 26 students, was randomly selected by the researcher to serve as either control or experimental groups.

4. Within the experimental group sections, the students were grouped randomly by the researcher into pairs, without regard to ability or gender.

5. During a regular geometry class period, prior to the start of the study, the classroom teacher demonstrated some of the basic features of *The Geometer's Sketchpad* program using a computer connected to a 32" television. The students were informed of the versatility and usefulness of the program for their study of geometry. They were also informed that they would be attending a three day training session on how to use the program, the first two days to learn the basics of using the program and the third day to assess what they had learned. Also, they were informed that they would need to know how to use the program in order to complete future in and out-of-class geometry assignments.

6. The students were informed, on a day prior to the start of the study, how they would be arranged in the computer lab for the purpose of working through the tutorial

materials. The participants in the two experimental groups (sections) were also informed how they were expected to work together and communicate with their partner during the two days of training. In particular, they were told that the control group would have each student working through the tutorial materials individually at a separate computer. The first experimental group section would have paired students working together cooperatively, with each pair at a single computer, to learn to use the software package. The second experimental group section would also have paired students working together cooperatively, but with each student in the pair having their own computer.

Students in the control group were instructed to work alone and not talk to other students in the classroom. Students in the two experimental groups were instructed to work through the tutorial materials together at the same rate of speed, speaking out loud to one another, and making sure that both of them understand how to do things. If the students would be sharing a single computer, they were also instructed to take turns controlling the keyboard and mouse.

All students were instructed that during the third day of the study, each student in all three groups would work silently and individually, at separate computers to complete an assessment tool to determine what they had learned during the previous two sessions.

Treatment

In order to conduct a controlled research study, the following steps were carried out during the actual treatment of the experimental groups and subsequent assessment of the learning that took place.

1. The regular classroom teacher, who taught all three sections of geometry, conducted the training sessions. This fact helped to assure continuity from section to section. The classroom teacher was familiar with the use of *The Geometer's Sketchpad* software, having used it the prior year in the classroom and having attended several workshops on its use.

2. The study was carried out over three consecutive days in the computer lab during each student's regular 50 minute geometry class period. During the first two days of the experiment, students worked through two separate tutorial lessons on learning to use *The Geometer's Sketchpad* program. On the third day, the students worked individually at separate computers to complete an assessment activity.

3. At the beginning of the two training sessions, the teacher checked that the students were sitting in their appropriate seats and then reminded them how they were to work through the tutorial materials, either silently by themselves or as cooperative pairs communicating with each other.

4. The classroom teacher provided each student with written tutorial materials, purchased from the publisher of the software, to work through in order to learn how to use the software.

5. The students were told when to begin working on the tutorial materials and to record the starting time on a provided sheet of paper. As each pair of students or individual student completed the tutorial material, they wrote their ending time, to the nearest minute, on the same sheet of paper. The tutorials were designed to be completed in about 30 minutes, assuring that each student would have plenty of time to complete the exercise during the 50 minute class period. Those students who finish the activity early

were allowed to work on other materials, either on or off the computer, but were not allowed to experiment freely with the software program. Also, they were not allowed to leave the testing room before the end of the class period.

6. Students were allowed to write down any notes they wanted to during the training sessions, notes which they could use during the assessment activity on the third day. During the assessment activity, however, the students were not allowed to use the tutorial sheets used during the first two days of training, nor were they allowed to talk to one another.

7. During the training sessions, the teacher circulated around the room observing the students, encouraging them in their work, and answering individual questions.

Post-Test Procedures

On the third day of the research study, the classroom teacher administered the assessment activity to all students that participated in the study to measure if mastery of the basics of *The Geometer's Sketchpad* program have been achieved.

As mentioned before, the post-test instrument was designed by the researcher and validated by the company which produced *The Geometer's Sketchpad* program. The post-test instrument consisted of two activities, corresponding to the two training sessions, which the students completed individually while working at separate computers. The students recorded their starting and ending times, to the nearest minute, for each activity and then saved their work on the computer network. It should be noted that the starting and ending time for the assessment activities were not used in the analysis of data because the researcher was interested in how long the students spend on learning the

materials during the training sessions during the first two days of the research study, not in how quickly they could complete the assessment activity. However, the students were still asked to record their times so the students would feel a need to work on the activity in a timely fashion. After completing the two activities and recording their ending times, the students emailed their results to the classroom teacher. The researcher then retrieved the email from the teacher and printed out the assessment activities for each student for further evaluation.

During the assessment sessions, the classroom teacher circulated around the room to observe the students working, but did not assist any students in completing the assessment activities.

Procedure for Data Analysis

Following the conclusion of the experiment, the amount of time needed to complete the tutorials and the number of students who successfully mastered the basics of the software package, based on their completion of the assessment activities, was tallied. In order to ensure greater consistency when analyzing the assessment activities, an assessment rubric was developed (see appendix C).

Each of the two post-assessment activities was first graded on a 10 point system, with 10 points indicating a perfect score. When the various steps in the activity were graded, half-points were deducted for slight errors. The two assessment activities that each student participated in were graded separately and then combined to produce a single assessment score for each student.

In order to determine if the use of cooperative pairs during the training process had any significant effect on the percentage of students who successfully learned to use the basics of the software package, a statistical test was performed. Because the study had one independent variable (group type), with three levels (the control group and two experimental groups), and one dependent variable (assessment score), with ratio level data measured in points, a one-way analysis of variance (ANOVA), with a pre-stated alpha value of 0.05, was used to check for statistical significance between groups. Because of the use of two separate post-assessment activities, corresponding to the two training sessions, separate ANOVAs were performed on the data obtained from each assessment activity.

The amount of time used during the tutorial activities was also analyzed to determine if there was a significant difference between the control group and the two experimental groups. A one-way ANOVA, with a pre-stated alpha value of 0.05, was used as the statistical test because there was one independent variable (group type), with three levels (the control group and two experimental groups), and one dependent variable (time), with ratio level data measured in minutes.

Expected Results

Based on the review of literature, especially as it pertained to the positive results associated with using cooperative groups in many other aspects of learning, it was expected that the use of cooperative pairs while learning to use new software programs would also produce favorable results. In particular, it was expected that students who

utilized cooperative pairs while learning to use a new software program would be significantly more likely to master the basics of the program's use than students who independently learned to use the program. It was also expected that students who utilized cooperative pairs while learning to use a new software program would learn to use the program significantly faster than students who independently learned to use the program.

Generalizability of Results

As explained earlier, the school in which the study was conducted was a parochial high school which limited entrance based on various criteria, such as a satisfactory score on a mathematics and English screening test, and a willingness to consider careers as pastors or teachers in the Wisconsin Evangelical Lutheran Synod's churches or schools. Also, the high school provided dormitories for approximately two-thirds of its students, with the remaining students commuting daily from home.

Because of the unique purpose, environment, and student body of the high school, even though it had a traditional four year mathematics sequence of algebra I, geometry, algebra II - trigonometry, and pre-calculus, generalizations to other high school populations may be limited.

Also, only one software program, *The Geometer's Sketchpad*, was used in the study. Generalizations to other software programs may be limited, even within the high school where the research was conducted.

Chapter IV

Results

This chapter presents the research data which was collected during the research study. The techniques which were used to analyze the research data are discussed, followed by the results which were obtained through the analysis.

Data Analysis

Following the conclusion of the research experiment, an analysis of the collected data was performed. This section on data analysis is divided into two parts, the first dealing with the amount of time participants spent on the tutorials, and the second examining the data as it pertains to the completion of the assessment activities.

Time Spent on Tutorials

As the research participants began and completed each of the two tutorials, they recorded their starting and ending times. (See appendix D for a listing of the specific times recorded by each of the participants.) Participants in the control group spent an average of 23.3 minutes per training session, while participants in experimental groups 1 and 2 spent an average of 19.7 minutes and 22.6 minutes, respectively.

After tabulating the data, the amount of time used during the tutorial activities was analyzed to determine if there was a significant difference between the control group and the two experimental groups. Because the data included one independent variable (group type), with three levels (the control group and two experimental groups), and one dependent variable (time), with ratio level data measured in minutes, a one-way analysis of variance (ANOVA), utilizing the pre-set alpha value of 0.05, was used as the statistical test. The results of the data analysis are presented in the next major section on "Findings."

Assessment Activity

This section explains the techniques used to analyze the data obtained from the assessment activities that the participants completed on the third day of the experiment. Two non-overlapping assessment activities were given, one corresponding to each of the two training sessions. In order to ensure greater consistency when analyzing the assessment activities, an assessment rubric was developed (see appendix C), and guidelines, as presented in the previous chapter, were developed for grading.

Utilizing these preset guidelines, each of the two assessment activities were graded on a 10 point scale, with 10 points indicating a perfect score. Half-points were deducted for slight errors. (See appendix E for a listing of the specific scores awarded to each participant.) The two assessment activities for each student were graded separately and then combined by adding them together to produce a single assessment score for each student.

Appendix E indicates that the average assessment score on activity 1 was 9.11, out of a possible 10, for the control group, and 8.75 and 8.76, respectively, for the two experimental groups. The average assessment score on activity 2 was 8.41 for the control group, and 9.00 and 8.48, respectively, for the two experimental groups. The combined assessment scores were very similar to one another between groups with an average of 17.52 for the control group, and 17.75 and 17.24, respectively, for the two experimental groups.

In order to determine if the use of cooperative pairs during the training process had any significant effect on the assessment scores of students, a statistical test was performed. Because the study had one independent variable (group type), with three levels (the control group and two experimental groups), and one dependent variable (assessment score), with ratio level data measured in points, a one-way ANOVA, utilizing the pre-set alpha value of 0.05, was used as the statistical test. Because of the use of two non-overlapping assessment activities, it was decided to perform a separate ANOVA on each of the two sets of assessment activity scores. The results of the data analysis are presented in the next section.

Findings

The findings are divided into two parts, the first dealing with the average amount of time each group spent working on the tutorials, and the second with the results of the assessment activity.

Time Spent on Tutorials

Appendix D indicates that the average amount of time each student in the control group spent working on the two training sessions varied from 17.5 minutes to 35.5 minutes, with an average time of 23.3 minutes. The appendix also indicates that students in experimental group 1 spent from 16.0 minutes to 29.0 minutes, with a group average of 19.7 minutes, and experimental group 2 spent from 16.5 minutes to 32.0 minutes, with a group average of 22.6 minutes. An examination of these group averages revealed that the two experimental groups did spend slightly less time in training, 3.6 minutes and 0.7 minutes respectively, than the control group. By visual inspection of the data in appendix D, it is obvious that the amount of time spent on training by various students within each of the three groups fluctuated greatly.

After comparing the mean, median, and mode of the dependent variable, time, (Table 1) it was assumed that the dependent variable was normally distributed. Next, a one-way ANOVA statistical test was performed on the data which resulted in a probability, or *p*-value, of 0.00647 (Table 2). This *p*-value indicated that there was a statistically significant difference between at least two of the group means.

Table 1

Measures of Central Tendency for Time Spent in Training

	Control	Exper. 1	Exper.2
Mean	23.348	19.667	22.577
Median	22.0	18.5	21.5
Mode	20.0	18.5	25.5

Note. All training times were measured in minutes.

The difference between group means for the control group and the two experimental groups was found to be 3.681 and 0.771 minutes, respectively. The difference between the group means of the two experimental groups was 2.910 minutes. In order to determine between which groups the significant difference existed, the Tukey's honestly significant difference (HSD) procedure (Tukey *a*) was performed. The results of the Tukey *a* procedure indicated that the significant difference in time spent on training occurred between the control group and experimental group 1, with a confidence interval of 0.833 to 6.529, and between experimental groups 1 and 2, with a confidence interval of -5.673 to -0.148. The next chapter takes a closer look at these significant differences, why they may have occurred, and their implication towards the future training of students to use new software programs.

Table 2

Analysis of Variance for Time Spent in Training

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Control	23	537	23.3478	23.0099
Exper. 1	24	472	19.6667	13.2754
Exper. 2	26	587	22.5769	13.9538

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	179.7333	2	89.8666	5.4211	0.00647	3.12768
Within Groups	1160.3969	70	16.5771			
Total	1340.1301	72				

Note. All training times were measured in minutes.

Assessment Activity

This section elaborates on the findings which resulted from an analysis of the data obtained from the assessment activities. From appendix E it can be observed that the average assessment score of 9.11 for the control group on assessment activity 1 was slightly higher than those of the two experimental groups, recorded as 8.75 and 8.76, respectively. But, just the opposite occurs when examining the data for assessment activity 2, where the average score of 8.41 for the participants in the control group is slightly lower than the average scores for those in the two experimental groups, at 9.00 and 8.48, respectively. In order to determine if the differences in assessment scores between groups was statistically significant, an ANOVA statistical test was performed.

After comparing the mean, median, and mode of the dependent variable, assessment score, for each of the assessment activities (Table 3) it was assumed that the dependent variable was normally distributed.

Table 3

Measures of Central Tendency for Assessment Activity Scores

	Assessment Activity 1			**Assessment Activity 2**		
	Control	*Exper. 1*	*Exper.2*	*Control*	*Exper. 1*	*Exper.2*
Mean	9.1136	8.7500	8.7619	8.4091	9.0000	8.4762
Median	9.0	9.0	9.0	9.25	9.5	9.0
Mode	10.0	9.0	9.0	10.0	10.0	10.0

Note. Scores were measured on a scale of 1 to 10.

A separate one-way ANOVA statistical test was then performed on each of assessment activities (Tables 4 and 5). The resulting probabilities or *p*-values for assessment activity 1 and activity 2 were 0.35892 and 0.38945, respectively. These results indicate that the assessment scores on activity 1 and 2 could have occurred strictly by chance 35.9% and 38.9% of the time, respectively.

Table 4

Analysis of Variance for Assessment Activity 1

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Control	22	200.5	9.1136	1.1174
Exper. 1	24	210	8.7500	0.9783
Exper. 2	21	184	8.7619	0.6155

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1.8962	2	0.9481	1.04124	0.35892	3.14044
Within Groups	58.2754	64	0.9106			
Total	60.1716	66				

Note. Assessment scores were measured on a scale of 1 to 10.

Table 5

Analysis of Variance for Assessment Activity 2

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Control	22	185	8.4091	3.4437
Exper. 1	24	216	9.0000	1.1087
Exper. 2	21	178	8.4762	3.2119

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	4.8467	2	2.42335	0.9570	0.38945	3.14044
Within Groups	162.0563	64	2.53213			
Total	166.9030	66				

Note. Assessment scores were measured on a scale of 1 to 10.

These *p*-values indicate that no statistically significant difference exists between the assessment scores of the three groups. This means that the null hypothesis, that the use of cooperative pairs during the training process will make no difference in the mastery of the basics of program use, can not be rejected. In other words, the research data collected does not show that there is a statistically significant advantage to using cooperative pairs during the training process in comparison to having students work individually through the tutorial training material.

By visual inspection of the data in appendix E, it was obvious that there was a greater fluctuation in scores within each of the three groups on assessment activity 2 than on activity 1. It was also obvious that a large number of individual scores on assessment activity 2 were lower than the scores on assessment activity 1. Overall, among the three

groups, 27 students out of 67, or 40%, scored lower on assessment activity 2 than they did on activity 1, 24 or 36% had the same score, and 16 or 24% achieved a higher score. If the 27 students with lower scores on assessment activity 2 are examined by group, the control group had 10 or 45% of its 22 students do poorer on assessment activity 2, followed by the two experimental groups with 10 or 42% and 7 or 33%, respectively.

The fact that 40% of the students had a lower score on assessment activity 2 is not as surprising as someone might first think for several reasons. First, these are two separate post-assessment activities, not a pre-test and a post-test where an increase in performance would be expected. Each assessment activity was testing the student's learning on different features of the computer program as presented during training sessions 1 and 2. The two assessment activities were never devised to be of the same difficulty, but rather to reliably test the student's understanding of the material presented in each specific training session. The first training session covered a number of simple program commands which were relatively easy to learn, where as, the second training session covered program features of greater difficulty. Therefore, it is not surprising that 40% of the students had a lower score on assessment activity 2.

Secondly, if a student had difficulty grasping the concepts presented in the first training session, as indicated by their score on assessment activity 1, it is only reasonable to expect that they might have even greater difficulty with the concepts presented in the more challenging second training session. One notable exception to this is found in the scores of student number 4 from experimental group 1. This student had a score of 6.5 on the first assessment activity and a 10 on the second (see appendix E). Apparently, what the student did was erase the circle instead of hide the circle in step 6 (see appendix A),

which caused the student to lose 2 points for step 3 as well. It does not appear that any of the other students committed this same kind of error.

After closely scrutinizing the assessment activity results of the 27 students who experienced a decrease in their assessment score from activity 1 to activity 2, there was no single question or set of questions on the assessment activities which caused the majority of these students to do poorly. It appears that all reductions in scores by these students are attributable to the fact that the students simply did not properly learn how to perform one or more functions of the software program or made some other simple mistake. While the geometry terminology used in the assessment activities, such as radius, median, and centroid, may have been confusing for some students, it was the exact same terminology as used in the training sessions, and therefore should have been known by the students.

It is also interesting to note that a larger percentage of the students in the experimental groups had total assessment activity scores of 16 or above in comparison to students in the control group (see appendix E). The number 16, out of a possible 20, was used as a cut-off point because it represents a score of 80%, or a C average. A score of 80% was often used at the school where the research was conducted to indicate an acceptable performance on other assessment activities.

The control group had 15 out of 22, or 68.2% of its students perform at the acceptable mark of 16 points or above. The students in experimental groups 1 and 2 had 21 or 87.5%, and 16 or 76.2%, respectively, of its students score at 16 points or above. These numbers indicate that the two experimental groups saw 19.3% and 8.0%, respectively, more of their students perform at or above the acceptable mark of 16 points

than the control group's students. Although these numbers are not statistically significant, it is interesting that the students in experimental group 1, where the students worked in cooperative pairs sharing a single computer, had the highest percentage of students who achieved this mark of acceptable performance.

Summary of Results

The research has shown that as students work through training tutorials on how to use a new software program, there is a statistically significant time savings for students who work together in cooperative pairs sharing a single computer, as compared to students who either work independently of one another or who work together in cooperative pairs with each student at a separate computer.

The research failed to show that students who work through training tutorials on how to use a new software program will perform significantly better on post-assessment activities than students who work through the training tutorials independently. However, the research did show that students who worked through the learning tutorials in cooperative pairs had post-assessment scores at least as high as students who worked through the materials independently.

Chapter V

Summary, Conclusions, Implications, and Recommendations

This chapter presents a summary of the entire research project, followed by conclusions which can be drawn from the research, as well as implications the research has for the future training of students using new software programs. Recommendations are also made for further research in the area of using cooperative pairs while training students to use new software programs.

Summary

Teachers often lament the fact that they need to spend valuable classroom time training students to use new software programs which will be needed as a tool by students to complete later course work (Schatz, 1996). Also, after completing the training process, too many students still have not mastered the basics of how to use the software program effectively, which means that the teacher then needs to use more classroom time for retraining. The purpose of this research study was to increase understanding of how the use of cooperative pairs during software training sessions would affect both the mastery of the basics of program use, as well as the amount of time needed to learn the basics. The specific research questions were:

1. Do students who cooperatively learn to use a software program, learn to use the program significantly faster than students who independently learn to use the same software program?

2. Are students who cooperatively learn to use a software program significantly more likely to master the basics of the program's use than students who independently learn to use the same software program?

Past research has shown that computers and electronic media can be used as an effective tool for the enhancement of the learning process in the classroom (Molnar, 1997; Hawkins, 1997). As the use of computers, as well as the sophistication of computer interfaces, has become more integrated into society in recent years, the need for training in the proper and efficient use of computers and software programs has become more critical (Baecker, Grudin, Buxton, & Greenberg, 1995). It is an unrealistic expectation that a user can simply begin using a new program effectively and efficiently without first spending some time learning the basics of how the program functions and how the user can best make use of the program's features.

The use of cooperative learning has become very common in classrooms in recent years (Strommen, 1995). Researchers, such as Johnson and Johnson (1994), have found that working in small cooperative groups has a positive impact on student achievement. However, a review of currently available research and literature has shed very little light on the use of cooperative groups during the software training process. In fact, research on procedures for training students to use new software programs appears to be very sparse. At the same time, computer usage as a tool in the classroom is becoming more prevalent than ever. But, in order for students to effectively use technology in the classroom, and

apply it as a tool to enhance learning, they must first spend time mastering the fundamentals of the technology itself.

The current research study was conducted at a parochial high school in Michigan over a three day period. Participants in the study were 75 10th grade geometry students learning to use the software package, *The Geometer's Sketchpad*, by Key Curriculum Press. The students were randomly scheduled into three geometry sections of between 24 and 26 students each, and the three sections were then randomly selected to serve as either control or experimental groups. The students then spent two class sessions working through training materials. The students in the control group worked through the training materials independently of one another, with each student at a separate computer. The students in the two experimental groups worked through the training materials in randomly assigned cooperative pairs. In one experimental group each cooperative pair was given a single computer to be shared by the two students, while in the other experimental group each student in each cooperative pair was given a separate computer. On the third day of the research study each student in all three groups independently completed two assessment activities to measure their mastery of the materials presented during the two training sessions.

The average amount of time spent by various students working through the training materials varied greatly within all three groups, from 16.0 minutes to 35.5 minutes. The average amount of time spent per training session for the control group and the two experimental groups was 23.3, 19.7, and 22.6 minutes, respectively. The only group where a noticeable reduction in time was observed was the experimental group in which the two students in each cooperative pair were required to share a single computer.

This result is attributable to the fact that sharing a single computer caused the students to stay more focused on their task than students in the other two groups. This time savings of approximately 3.6 minutes per training session was found to be statistically significant with a probability or *p*-value of 0.00647 of having occurred by chance.

Two post-assessment activities, corresponding to the two training sessions, were completed by 67 of the original 75 students on the third day of the study. Each of the two assessment activities were analyzed separately. Although the analysis found that there was no statistically significant advantage to using cooperative pairs over independent learning, it did show that the independent learning group did slightly better on the first assessment activity and that the cooperative pair group, with one computer per pair, did slightly better on the second assessment activity. The second activity consisted of more difficult procedures than the first. It is questionable if the two assessment activities were sensitive enough to detect small variations in learning given the fact that nearly 70% of the participants in the study scored a 9 or above, out of a possible 10 points, on each of the assessment activities.

It is interesting to note that when the number of students who attained a score of 8 or more points on the assessment activities was examined, 8 representing a score of 80% which was the score often used by the school where the research was conducted to indicate a satisfactory or acceptable performance, the students who used cooperative pairs sharing a single computer outperformed the control gr ›up students by 19.3%. Specifically, the data reveals that 68.2% of the participants in the control group satisfactorily completed the assessment activities in comparison to 87.5% and 76.2 % for the participants in the two experimental groups. From these statistics it can also be seen

that the students who worked in cooperative pairs with one computer per pair performed slightly better than those students who worked in cooperative pairs where each student had a separate computer, 87.5% to 76.2%, respectively. Although these results may appear to be significant to most educators from a practical point of view, they are not statistically significant.

In summary, this research study showed that the use of cooperative pairs, with each pair sharing a single computer, during the training process of learning how to use a new software program was beneficial in that it reduced the amount of time students spend in training. However, further study is still recommended. If the use of cooperative pairs during the training process can be shown to increase the number of students who successfully master the basics of program use, at a statistically significant level, then schools may need to change their current training practices. This means that schools may want to move away from the traditional training method of one student per computer, independently working through the training materials, to a method of utilizing cooperative pairs, with each pair sharing an individual computer, during the training process.

As suggested above, more research in the area of using cooperative pairs during the training process needs to be completed. Such research should be carried out with larger sample sizes and should concentrate on two groups, a control group, with one computer per student working through the training materials independently, and an experimental group in which students work in cooperative pairs with a single computer for each pair.

Conclusions

The purpose of the study was to increase understanding of how the use of cooperative pairs, while learning to use a new software program, would affect both the mastery of the basics of program use as well as the length of time needed to learn the basics.

The research did support the hypothesis that students who cooperatively learn to use a new software program will learn to use the program faster than students who independently learn to use the same program. The research data revealed an average time savings per training session of 3.7 minutes per participant between the control group, spending an average of 23.3 minutes per participant during training, and experimental group 1, whose participants spent an average of 19.7 minutes per training session. The control group participants worked through the training materials independently at separate computers while the participants in experimental group 1 worked in cooperative pairs with each pair sharing a single computer. The research also revealed a statistically significant time savings of 2.9 minutes between experimental group 2, working in cooperative pairs with each student having an individual computer, and experimental group 1. The participants from experimental group 2 saw a slight, 0.8 minute, non-significant reduction in time spent in training when compared to the control group.

The 3.7 minute average reduction in time experienced by experimental group 1 can be attributed to the fact that students who are forced to work together at a single computer are more likely to keep one another on task, thus reducing the overall time needed to complete a particular training exercise. Students in experimental group 2,

which only experienced a slight reduction in training time, 0.8 minutes, were not as likely to keep one another on task because each student was viewing their own monitor rather than sharing it with the other student in the cooperative pair.

The fact that experimental group 1, which consisted of students working in cooperative pairs with each pair sharing a single computer, saw a statistically significant decrease in time needed to complete the training sessions clearly supports what social constructivists would predict. A major theme of social constructivism, as expressed in the theoretical framework established by Vygotsky, as well as in the publications of the NCTM, is that social interaction plays a fundamental role in the development of cognition (Kearsley, 1998). Therefore, to a social constructivist it would not be surprising that students who worked together in cooperative pairs would be able to complete tasks more quickly than students working alone. Also, students in cooperative pairs who shared a single computer would experience an higher degree of social interaction and thus see an even greater reduction in time needed to complete a task than those students who worked in cooperative pairs with an individual computer for each student.

Concerning learning to use new software, the research does not allow one to conclude that students who cooperatively learn to use a new software program will be significantly more likely to master the basics of the program's use than students who independently learn to use the same program. In fact, the research indicates that the training method used, individual or some form of cooperative pairs, had little or no effect on mastery of the basics of the program's functions.

The fact that no discernable difference in learning based on the method of training was indicated by the assessment activities may be attributable to several factors. First, the research data did note a reduction in assessment scores for the control group from post-assessment activity 1 when compared to activity 2, but just the opposite occurred for the two experimental groups, which saw a slight increase in assessment scores between activity 1 and 2. The materials presented during the first training session, and then assessed by assessment activity 1, may have been so simple and basic that the particular method of training would have had no effect on the amount of learning which took place. The materials presented during the second training session, and then assessed by assessment activity 2, were of a slightly more difficult nature, and the data results indicated that experimental group 1 did slightly, but not significantly, better. There are two more training sessions produced by Key Curriculum Press, which delve into more complex features of the software program, but which were not used in the study because they go beyond the basics that the students needed for their geometry class. It is possible that if these additional training sessions had been used and the resulting learning assessed, that a statistically significant difference would have been found between the control group and the two experimental groups. In other words, it may be that the difficultly level of the material being learned had an effect on the research results, or in the case of this research, that the lack of difficulty of the basic features of the software program being learned caused the differences in assessment scores between groups to be statistically insignificant.

A second limitation of the study may be that the assessment activities used, although an indicator that learning took place, were not sensitive enough to accurately

measure slight variations in learning. The fact that nearly 70% of the participants in the study scored a 9 or above, out of a possible 10 points, on assessment activity 1 and nearly 65% of the participants did the same on assessment activity 2, indicates that the two post-assessment activities did not do a good job of separating or stratifying the participants according to the amount of learning achieved.

Another limitation of this research study is its small sample sizes. McMillan and Schumacher (1993) point out that sample size is an important factor in obtaining statistically significant results, especially when looking at small differences or slight relationships, which is often the case when looking at score improvements on assessment activities. McMillan and Schumacher suggest, that in such cases, the larger the size of the sample the better.

Implications

This study shows that the use of cooperative groups during the training process to learn how to use new software programs is beneficial from a time savings perspective, but needs further study concerning whether or not the use of cooperative pairs increases learning of the program basics. One clear conclusion from the study was that the particular way in which cooperative pairs are used during the training process effects the time savings which results. Cooperative pairs that shared a single computer had a significant time savings when compared to cooperative pairs where each student had their own computer.

Although a savings in the amount of time needed by students to complete the training process can be achieved through the use of cooperative pairs, the real benefit of using cooperative pairs may still be in the area of mastering the basics of how to use the program. Such mastery would result in a time savings in the long run because teachers would not need to spend as much time retraining students who did not master the basics during the initial training sessions.

There is an obvious need for more research in the use of cooperative pairs during the training process of learning how to use new software programs. If the use of cooperative pairs during the training process can be shown to increase the number of students who successfully master the basics of program use, at a statistically significant level, then schools may need to change their current training practices, which often involve one student per computer working through the training materials independently of one another. Also, there appears to be no advantage to having a single computer for each student during the training process in comparison to having two students per computer, in fact, this study shows that there is an advantage to having two students share the same computer in order to reduce the amount of time needed to complete the training materials.

Recommendations

As mentioned above, more research in this area is recommended before schools drastically change their current training formats. The one exception, however, is that schools which are currently using cooperative pairs while training students to use new software should consider having the pair share a computer. This will reduce the amount

of time which the pair spends on training and will not reduce the amount of learning which takes place.

The main recommendation for future research in this area is to modify the study so that it contains only two groups, a control group, with one computer per student, each working independently of one another, and an experimental group, made up of cooperative pairs working together with a single computer for the pair. Such an arrangement would help to focus on the main issue, the use of cooperative pairs verses independent training. A second recommendation would be to develop post-assessment activities which more acutely measure slight changes in learning in order to better separate or stratify each participant's results based on his or her actual level of learning. A final recommendation would be to dramatically increase the sample sizes of each group, which would also help to detect slight, but important, increases in learning.

In conclusion, as more and more students begin to spend more time learning to use new software programs, research in effective training methods for students becomes more important. The use of cooperative pairs during the training process of learning how to use new software programs may be advantageous, but more research is needed on the subject before wholesale revisions of software training methods can be recommended to schools.

Appendix A

Assessment Activity

Section:___ Computer #_____

Sketchpad Activities

Name _____

Please work through the following activities as best you can by yourself. Please write down the starting time and ending time for each activity to the nearest minute.

Activity 1: Starting time: _____

1. Create a point.
2. Starting at the point constructed in step 1, draw a segment.
3. Now, construct a circle with the segment from step 2 serving as the radius of the circle.
4. Construct another radius for the circle.
5. Connect the endpoints of the radii to complete a triangle.
6. Hide the circle so only the triangle is visible.
7. Label the 3 vertices of the triangle as E, F, and G.
8. Save your sketch in your directory on the f: drive as **f:activ1.gsp**

Ending time: _____

Activity 2: Starting time: _____

1. Use the segment tool to construct a triangle.
2. Construct a midpoint on each of the three sides of the triangle.
3. Construct just two of the triangle's three medians.
4. Construct a point at the intersection of the two medians.
5. Label the point in step 4 as **Centroid**.
6. Construct the third median.
7. Use the Label tool to display the labels of the endpoints of one of the medians.
8. Measure the distance from the labeled vertex to the centroid.
9. Measure the distance from the labeled midpoint to the centroid.
10. Calculate the ratio of the first measurement to the second.
11. Save your sketch in your directory on the f: drive as **f:activ2.gsp**

Ending time: _____

Appendix B

Validation Letter for Assessment Activity

Dan Bennett
Key Curriculum Press
Berkeley, CA 94702
Fax: (510) 548-0755
September 29, 1998

Prof. James R. Grunwald
Director of Academic Computing
Martin Luther College
1995 Luther Court
New Ulm, MN 56073-3965

Dear James,

I have inspected the attached "Appendix A: Assessment Activity" which you sent to me for my validation. I have found that the items which you are asking on the assessment activity do cover the topics which users would learn to do when working through "Guided Tour I: The Freehand Tools" and "Guided Tour II: The Centroid of a Triangle" as found in the book "Exploring Geometry with the Geometer's Sketchpad," published by Key Curriculum Press and authored by me. Your assessment activity is valid and should provide reliable results concerning how well the students have mastered the basics of using the Geometer's Sketchpad program as presented in the two Guided Tours mentioned above.

Sincerely,

Dan Bennett
Key Curriculum Press
dbennett @keypress.com

Appendix C

Assessment Rubric

Section:___ Computer #_____

Sketchpad Activities

Name _____

Please work through the following activities as best you can by yourself. Please write down the starting time and ending time for each activity to the nearest minute.

Activity 1: Starting time: _____
Total of 10 points possible.

1. Create a point.
   1 point
2. Starting at the point constructed in step 1, draw a segment.
   1 point
3. Now, construct a circle with the segment from step 2 serving as the radius of the circle.
   2 points (1 point for a circle, 2nd point is a segment forms the radius)
4. Construct another radius for the circle.
   1 point (end points must be attached to the circle)
5. Connect the endpoints of the radii to complete a triangle.
   1 point (triangle made of three connected segments)
6. Hide the circle so only the triangle is visible.
   1 point
7. Label the 3 vertices of the triangle as E, F, and G.
   1 point (½ point if only 1 or two labeled properly)
8. Save your sketch in your directory on the f: drive as **f:activ1.gsp**
   1 point

Ending time: _____

1 point (for not having more than 1 extra stray point, segment, etc., on final sketch)

Activity 2: Starting time: _____
Total of 10 points possible.

1. Use the segment tool to construct a triangle.
   1 point
2. Construct a midpoint on each of the three sides of the triangle.
   1 point (each midpoint must be constructed, not approximated)
3. Construct just two of the triangle's three medians.
   1 point (must be medians, not mid-segments connecting midpoints)
4. Construct a point at the intersection of the two medians.
   1 point (if step three connected midpoints, then this step will also be wrong)
5. Label the point in step 4 as **Centroid**.
   1 point (½ point if labeled, but not as Centroid)
6. Construct the third median.
   1 point (not considered wrong if mid-segments constructed in step 3)
7. Use the Label tool to display the labels of the endpoints of one of the medians.
   1 point (½ point if only 1 endpoint labeled, also, no deduction if other endpoints also labeled)
8. Measure the distance from the labeled vertex to the centroid.
   1 point (½ point if any distance was measured)
9. Measure the distance from the labeled midpoint to the centroid.
   1 point (½ point if any distance was measured)
10. Calculate the ratio of the first measurement to the second.
    1 point (½ point if fixed ratio calculated, also, no deduction if inverse ratio used)
11. Save your sketch in your directory on the f: drive as **f:activ2.gsp**
    0 points (this is just a formality, no point awarded)

Ending time: _____

Appendix D

Results of Time Spent in Training

Results of Time Spent in Training											
Control Group				Experimental Group 1				Experimental Group 2			
Independent Training (One student per computer)				Cooperative Pairs (Two students per computer)				Cooperative Pairs (One student per computer)			
Student Id #	(23 students*) Time in Training			Student Id #	(24 students) Time in Training			Student Id #	(26 students) Time in Training		
	Day 1	Day 2	Ave.		Day 1	Day 2	Ave.		Day 1	Day 2	Ave.
1	15	25	20.0	1	14	21	17.5	1	20	26	23.0
2	38	26	32.0	2	14	21	17.5	2	20	26	23.0
3	21	29	25.0	3	30	28	29.0	3	18	24	21.0
4	20	22	21.0	4	30	28	29.0	4	18	24	21.0
5	20	20	20.0	5	18	20	19.0	5	15	18	16.5
6	18	22	20.0	6	18	20	19.0	6	15	18	16.5
7	25	29	27.0	7	16	21	18.5	7	20	21	20.5
9	18	18	18.0	8	16	21	18.5	8	20	21	20.5
10	23	24	23.5	9	12	25	18.5	9	28	36	32.0
11	18	24	21.0	10	12	25	18.5	10	28	36	32.0
12	20	18	19.0	11	12	20	16.0	11	25	26	25.5
13	32	25	28.5	12	12	20	16.0	12	25	26	25.5
15	17	22	19.5	13	13	27	20.0	13	23	20	21.5
16	21	25	23.0	14	13	27	20.0	14	23	20	21.5
17	22	22	22.0	15	27	20	23.5	15	17	21	19.0
18	32	39	35.5	16	27	20	23.5	16	17	21	19.0
19	20	15	17.5	17	17	20	18.5	17	20	28	24.0
20	29	28	28.5	18	17	20	18.5	18	20	28	24.0
21	25	23	24.0	19	24	21	22.5	19	17	23	20.0
22	25	15	20.0	20	24	21	22.5	20	17	23	20.0
23	20	19	19.5	21	15	18	16.5	21	25	26	25.5
24	30	30	30.0	22	15	18	16.5	22	25	26	25.5
25	20	25	22.5	23	20	13	16.5	23	21	26	23.5
				24	20	13	16.5	24	21	26	23.5
								25	19	24	21.5
								26	19	24	21.5
Ave.	23.0	23.7	23.3	Ave.	18.2	21.2	19.7	Ave.	20.6	24.5	22.6

***Note:** In the Control Group, 1 of the original 25 students did not complete the training due to absenteeism and 1 was excluded due to a learning disability.

Summary of results of time spent in training:

The average amount of time spent in training per student during each training session was 23.3 minutes for the control group, and 19.7 and 22.6 minutes, respectively, for each of the experimental groups.

Appendix E

Individual Assessment Activity Scores

Individual Assessment Activity Scores

Control Group: Independent Training (One student per computer)				Experimental Group 1: Cooperative Pairs (Two students per computer)				Experimental Group 2: Cooperative Pairs (One student per computer)			
Student Id #	(22 students*) Activity Number 1	2	Total	Student Id #	(24 students) Activity Number 1	2	Total	Student Id #	(21 students*) Activity Number 1	2	Total
1	10	10	20.0	1	9	6.5	15.5	1	9	7.5	16.5
2	10	10	20.0	2	8	7	15.0	2	10	9.5	19.5
4	9	7.5	16.5	3	10	9.5	19.5	3	7.5	9	16.5
5	9	5.5	14.5	4	6.5	10	16.5	4	8.5	6	14.5
6	9.5	6	15.5	5	7	9.5	16.5	5	9	10	19.0
7	9	10	19.0	6	7	8	15.0	6	8	9	17.0
9	9	9	18.0	7	10	9	19.0	7	9	9	18.0
10	10	10	20.0	8	9	7	16.0	8	10	10	20.0
11	10	9	19.0	9	9	9.5	18.5	10	8	7	15.0
12	8	6	14.0	10	9	8.5	17.5	11	10	10	20.0
13	9	5.5	14.5	11	10	9.5	19.5	12	9	9	18.0
15	10	8.5	18.5	12	7.5	9.5	17.0	14	8	5.5	13.5
16	10	10	20.0	13	9	10	19.0	15	10	10	20.0
17	10	10	20.0	14	10	10	20.0	16	9	9.5	18.5
18	7.5	8	15.5	15	8	9	17.0	17	9	10	19.0
19	9	9.5	18.5	16	9	10	19.0	18	9	10	19.0
20	7.5	5.5	13.0	17	10	8.5	18.5	21	8	8	16.0
21	10	10	20.0	18	8.5	9.5	18.0	22	8	9	17.0
22	10	9.5	19.5	19	9	10	19.0	23	8	4	12.0
23	6	5.5	11.5	20	8.5	10	18.5	24	8	6	14.0
24	9	10	19.0	21	9	9	18.0	25	9	10	19.0
25	9	10	19.0	22	9	8.5	17.5				
				23	9	10	19.0				
				24	9	8	17.0				
Ave.	9.11	8.41	17.52	Ave.	8.75	9.00	17.75	Ave.	8.76	8.48	17.24

***Note:** In the Control Group, 3 of the original 25 students did not complete the study; 1 due to absenteeism, 1 for non-completion of the assessment activity, and 1 was excluded due to a learning disability. In Experimental Group 2, 5 of the original students did not complete the study; 3 due to absenteeism, and 2 for non-completion of the assessment activity.

Summary of results of time spent in training:

The table shows the score that each student obtained on each assessment activity, as well as the combined score, or total, on the two activities. The highest score possible on an individual assessment activity was a 10. The bottom row of the table indicates the group average for each of the assessment activities and the average of the combined total scores.

Reference List

Abelson, H. & diSessa, A. (1982). *Turtle geometry: The computer as a medium for exploring mathematics.* Cambridge, MA: MIT Press.

Alexander, T. E. (1991). *A study to examine the effects of computers and traditional teaching methods on 9-11 year old students learning to add and subtract fractions.* Unpublished doctoral dissertation, Nova University, Fort Lauderdale, FL.

Andleigh, P. H., & Thakrar, K. (1996). Multimedia authoring and user interface. In *Multimedia systems design* (pp. 433-470). Upper Saddle River, NJ: Prentice Hall.

Appleton, E. L. (1998). 1998 State of the industry report. *Inside Technology Training*, 2 (6), 12-18.

Atkins, N. E., & Vasu, E. S. (1998). The teaching with technology instrument: Effectively measuring where teachers are and planning for staff development. *Learning and Leading with Technology*, 25 (8), 35-39.

Baecker, R. M., Grudin, J., Buxton, W. A. S., & Greenberg, S. (1995). Designing to fit human capabilities. In *Human-computer interaction: Toward the year 2000* (pp. 667-680). San Francisco: Morgan Kaufmann Publications.

Bailey, G. D. (1997). What technology leaders need to know. *Learning and Leading with Technology*, 25 (1), 57-62.

Barksdale, J. M. (1996). New teachers: Unplugged - Why schools of education are still sending you staff you'll have to train in technology. *Electronic Learning,* 15 (5), 38-45.

Barnes, J. H. (1991). *An investigation of the effectiveness of computer-assisted mathematics instruction as opposed to traditional instruction.* Unpublished doctoral dissertation, Nova University, Fort Lauderdale, FL.

Bennett, D. (1996). *Exploring geometry with Geometer's Sketchpad* (pp. 34-40). Berkeley, CA: Key Curriculum Press.

Bohlen, G. A., & Ferratt, T. W. (1993). The effect of learning style and method of instruction on the achievement, efficiency and satisfaction of end-users learning computer software. In *SIGCPR '93. Proceedings of the 1993 conference on computer personnel research*, (pp. 273-283). St. Louis, MO.

Bork, A. (1997). The future of computers and learning. *T.H.E. Journal*, 24 (11), 69-77.

Boyd, S. (1998). 10 commandments for trainers. *Inside Technology Training*, 2 (5), 26-27.

Britt, C., Eurich-Fulcer, & Schofield, J. (1994). Teachers, computer tutors, and teaching: The artificial intelligent tutor as an agent for classroom change. *American Educational Research Journal*, 31 (3), 579-607.

Butler, K. A., Jacob, R. J. K., & John, B. E. (1998). Human-computer interaction: Introduction & overview. In *CHI '98. Proceedings of the CHI '98 summary conference: Human factors in computing systems*, (pp. 105-106).

Carroll, J. M., & Mack, R. L. (1995). Learning to use a word processor: By doing, by thinking, and by knowing. In *Human-computer interaction: Toward the year 2000* (pp. 698-717). San Francisco: Morgan Kaufmann Publications.

Casey, D., Jr. (1987). A comparison of student achievement using printed drill and practice materials and computer assisted instruction. In M. Miller (Ed.), *Seventh Annual Microcomputers in Education Conference: Who's in charge?* (pp. 36-43). Rockville, MD: Computer Science Press.

Chia, J., & Duthie, B. (1993). Primary children and computer-based art work: Their learning strategies and context. *Art Education*, 46 (6), 23-26, 35-41.

Chiu, J. W. K. (1995). A training selection model for Asian undergraduate students in office automation software: cooperative learning versus whole-group instruction. In *SIGCPR '95. Proceedings of the 1995 ACM SIGCPR conference on Supporting teams, groups, and learning inside and outside the IS function reinventing IS* (pp. 229-230). Nashville, TN.

Coburn, P., Kelman, P., Roberts, N., Snyder, T. F. F., Watt, D. H., & Weiner, C. (1985). *Practical guide to computers in education.* Reading: MA: Addison-Wesley.

Cuoco, A. A., Goldenberg, E. P., & Mark, J. (Eds.). (1994). A potpourri. *Mathematics Teacher*, 87 (7), 566-569.

DeCorte, E. (1992). On the learning and teaching of problem-solving skills in mathematics and Logo programming. *Applied Psychology: An International Review-Psyychologie Applique: Revue Internationale, 41* (4), 8-10.

Denning, R., & Smith, P. J. (1995). *Teaching problem-solving through a cooperative learning environment.* CHI'95, Denver, CO, ACM 0-89791-755-3/95/0005.

Dishon, D., & O'Leary, P. W. (1994). *A guidebook for cooperative learning: A technique for creating more effective schools* (2nd ed.). Holmes Beach, FL: Learning Publications.

Dockterman, D. A. (1997). My path to the one computer classroom. In *Great teaching in the one computer classroom* (4th ed., pp. 1-10). Watertown, MA: Tom Snyder Productions.

Dubinsky, E., & Schwingendorf, K. (1997). *Calculus, Concepts, Computers, and Cooperative Learning (C4L).* http://www.math.purdue.edu/~ccc/ Revised January 22, 1997. Accessed March 29, 1997. Author's email: edd@cs.gsu.edu & ks&math.purdue.edu

Dugdale, S., LeGare, O., Matthews, J. I., & Ju, M. (1998). Mathematical problem solving and computers: A study of learner-initiated application of technology in a general problem-solving context. *Journal of Research on Computing in Education*, 30 (3), 239-253.

Dyrli, O. E., & Kinnaman, D. E. (1994). Integrating technology into your classroom curriculum. *Technology and Learning,* 14 (5).

Dyrli, O. E., & Kinnaman, D. E. (1995). Developing a technology-powered curriculum. *Technology and Learning,* 15 (5), 46-51.

Eddins, S. K., Maxwell, E. O., & Stanislaus, F. (1994). Geometric transformations. *Mathematics Teacher,* 87 (3), 177-180.

Emmans, C. C., & Byxbee, W. E. (1997). Technology assessment and training for a Brazilian school. *T.H.E. Journal*, 24 (6), 55-57.

Enderson, M. (1997). Old problems, new questions - Using technology to enhance math education. *Leading & Learning with Technology*, 25 (2), 28-32.

Foster, A. G. (1993). *Cooperative learning in the mathematics classroom.* Columbus, OH: Glencoe, Macmillan/ McGraw-Hill Publications.

Gates, B. (1995). Education: The best investment. In *The road ahead* (pp. 184-204). New York: Viking Penguin.

Gist, M. E., Schwoerer, C., & Rosen, B. (1989). Effects of alternative training methods on self-efficacy and performance in computer software training. *Journal of Applied Psychology*, 74 (6), 884-891.

Griest, G. (1996). Computer education as an obstacle to integration and internetworking. *Learning and Leading With Technology*, 23 (7), 31-34.

Grunwald, J. R. (1990). The role of computers in education. *The Lutheran Educator*, 31 (1), 4-7.

Grunwald, J. R. (1997). *Introduction to HCI.* http://www.mlc-wels.edu/jgrunwald/introhci.html Revised September 3, 1998. Accessed September 5, 1998. Author's email: grunwajr@mlc-wels.edu Author's Web Site: http://www.mlc-wels.edu/jgrunwald/

Harrington-Lueker, D. (1996). Coming to grips with staff development. *Electronic Learning*, 16 (1), 32-43.

Hawkins, J. (1997). The world at your fingertips. In P. Burness & W. Snider (Eds.), *Learn & live* (pp. 213-215). Nicasio, CA: The George Lucas Educational Foundation.

Holzberg, C. S. (1998). Dive in! Don't worry, you'll swim! *Electronic Learning*, 17 (4).

Hoyles, C. & Noss, R. (1994). Dynamic geometry environments: What's the point? *Mathematics Teacher*, 87 (9), 716-717.

Jackiw, N. (1990). The Geometer's Sketchpad [Computer software]. Berkeley, CA: Key Curriculum Press.

Jewett, T. (1996). *A cooperative learning approach to teaching social issues of computing*. ACM 0-89791-827-4/96/02.

Johnson, D. W., & Johnson, R. T. (1989). *Cooperation and competition*. Edina, MN: Interaction Book.

Johnson, D. W., & Johnson, R. T. (1994). *Learning together and alone: Cooperative, competitive, and individualistic learning* (4th ed.). Boston, MA: Allyn and Bacon, Interaction Book.

Johnson, D. W., Johnson, R. T., & Smith, K. A. (1991). Active learning: Cooperation in the college classroom. Edina, MN: Interaction Book.

Kearsley, G. (1998). *Social development theory.* http://www.gwu.edu/~tip/vygotsky.html Accessed March 8, 1990. Author's email: kearsley@fcae.nova.edu Author's Web Site: http://www.gwu.edu/~tip/

Kelly, J. T., & Leckbee, R. (1998). Reality check: What do we really know about technology, and how do we know it? *Syllabus*, 12 (1), 24-26, 53.

Kennedy, J. (1996). It's unclear whether schools will use computers in most effective way. *R & D Watch*, 1 (3), 4.

Lewis, C., & Rieman, J. (1994). Getting to know users and their tasks. In R. M. Baecker, J. Grudin, W. A. S. Buxton, & S. Greenberg (Eds.), *Readings in human-computer interaction: Toward the year 2000* (1995), (pp. 122-127). San Francisco: Morgan Kaufmann Publications.

Liao, Y. C. (1998). Effects of hypermedia versus traditional instruction on students' achievement: A meta-analysis. *Journal of Research on Computing in Education*, 30 (4), 341-359.

Lovely, G. (1997a). After the workshop is over... Six effective ways to follow up any staff development session. *Electronic Learning*, 16 (5), 53.

Lovely, G. (1997b). Top 10 training blunders: Avoid these common missteps in technology professional development. *Electronic Learning*, 16 (6), 63.

Mack, R. L., & Nielsen, J. (1994). Executive summary. In J. Nielsen & R. L. Mack (Eds.), *Usability inspection methods* (pp. 1-23). New York: John Wiley & Sons.

McFarland, R. D. (1995). Ten design points for the human interface to instructional multimedia. *T.H.E. Journal*, 22 (7), 67-69.

McMahon, H. (1990). Collaborating with computers. *Journal of Computer Assisted Learning*, 6, 149-167.

McMillan, J. H., & Schumacher, S. (1993). Evaluation research and policy analysis. In *Research in education: A conceptual introduction* (3rd ed.), (pp. 518-560). New York: HarperCollins.

McSweeney, J. (1997). Technology in action. In P. Burness & W. Snider (Eds.), *Learn & live* (pp. 216-217). Nicasio, CA: The George Lucas Educational Foundation.

Merriam-Webster's collegiate dictionary (10th Ed.). (1995). Springfield, MA: Merriam-Webster.

Molnar, A. R. (1997). Computers in education: A brief history. *T.H.E. Journal*, 24 (11), 63-68.

Natasi, B. K., Battista, M. T., & Clements, D. (1990). Social-cognitive interactions, motivation, and cognitive growth in Logo programming and CAI problem-solving environments. *Journal of Educational Psychology*, 82 (21), 150-158.

National Council of Teachers of Mathematics. (1989). *Curriculum and evaluation standards for school mathematics*. Reston, VA: Author.

National Council of Teachers of Mathematics. (1991). *Professional standards for teaching mathematics*. Reston, VA: Author.

National Council of Teachers of Mathematics. (1995). *Assessment standards for school mathematics*. Reston, VA: Author.

Nielsen, J., & Mack R. L. (1994). *Usability inspection methods*. New York: John Wiley & Sons.

Nilsen, E., Jong, H., Olson, J. S., Biolsi, K., Rueter, H., & Mutter, S. (1993). The growth of software skill: A longitudinal look at learning & performance. In *CHI '93. Conference proceedings on Human factors in computing systems*, (pp. 149-156).

November, A. (1997). Magic links - Changing the focus of technology planning. *Learning and Leading with Technology*, 24 (8), 54-56.

O'Connor, V. (1997). Cooperative groups: What role do they play in the teaching and learning of mathematics? *Xchange*, 2 (1).

Olfman, L., & Bostrom, R. P. (1991). End-user software training: An experimental comparison of methods to enhance motivation. *Journal of Information Systems*, 1 (4), 249-266.

Olfman, L., & Mandviwalla, M. (1992). *An experimental analysis of end user software training manuals*. Manuscript submitted for publication.

O'Malley, C. (1992). Designing computer systems to support peer learning. *European Journal of Psychology of Education*, 8 (4), 339-352.

Pace, J., & Gardner, H. (1997). Building a bridge to knowledge for every child. In P. Burness & W. Snider (Eds.), *Learn & live* (pp. 17-21). Nicasio, CA: The George Lucas Educational Foundation.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: BasicBooks.

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. New York: BasicBooks.

Perelman, L. J. (1992). *School's out*. New York: Avon Books.

Ploeger, F. D., (1984). Instructional microcomputing: A survey of the Research. In C. D. Martin & R. S. Heller (Eds.), *Capitol-izing on computers in education, Proceedings of the 1984 Association of Educational Data Systems annual convention* (pp. 267-273). Rockville, MD: Computer Science Press.

Rasmussen, S. (1996, March). *Technology issues in teacher education.* Paper presented at the EXXON/Cal Poly Conference on Mathematics Teacher Education and Development, Pomona, CA.

Riedl, R. (1987). The case for the computer as a tool in the classroom. In M. Miller (Ed.), *Seventh Annual Microcomputers in Education Conference: Who's in charge?* (pp. 204-210). Rockville, MD: Computer Science Press.

Roschelle, J. (1994). Collaborative inquiry: Reflections on Dewey and learning technology. *The Computing Teacher*, 22 (8), 6-9.

Salisbury, D. (1985). *Using microcomputers for drill and practice: issues and implications.* Paper presented at the Annual Meeting of the Association for the Educational Communications and Technology: Anaheim, CA.

Saxton, M. (1995). Art, technology, and cooperative learning: Research and personal experience. *Learning and Leading with Technology*, 23 (2), 66-66.

Schatz, S. (1996). Show/do/cue: A model for training use of software tools. *T.H.E. Journal*, 24 (2), 86-89.

Serra, M. (1997). Michael Serra's suggestions on cooperative learning. In J. Bergez (Ed.), *Discovering geometry: Teacher's guide and answer key* (pp. 17-34). Berkeley, CA: Key Curriculum Press.

Shayo, C., & Olfman, L. (1993). Is the effectiveness of formal end-user software training a mirage? In *SIGCPR '93. Proceedings of the 1993 conference on computer personnel research,* (pp. 88-99). St. Louis, MO.

Shayo, C., & Olfman, L. (1994). A three dimensional view and research agenda for the study of transfer of skills gained from formal end-user software training. In *SIGCPR '94. Proceedings of the 1994 computer research conference on Reinventing IS: Managing information technology in changing organizations* (pp. 133-144). Alexandria, VA.

Siegel, J. (1995). The state of teacher training: The results of the first national survey of technology staff development in schools. *Electronic Learning,* 14 (8), 43-45, 48-52.

Silverman, B. G. (1995). Computer supported collaborative learning (CLS). *Computers Education*, 25 (3), 81-91.

Slavin, R. (1990). *Cooperative Learning: Theory, research and practice*. Des Moines, IA: Prentice Hall.

Smith, D., & Tarkow, J. (1998). Individualized learning: The self-paced computer lab. *T.H.E. Journal*, 25 (10), 62-64.

Strommen, E. (1995). Cooperative learning: Technology may be the Trojan horse that brings collaboration into the classroom. *Electronic Learning*, 14 (6), 24-28, 33-35.

Sullivan, P. S. (1994). Computer technology and collaborative learning. *New Directions for Teaching and Learning*, 59 (Fall 1994), 59-67.

Tally, B., & Grimaldi, C. (1995). Developmental training: Understanding the ways teachers learn. *Electronic Learning*, 14 (8), 14-15.

Tayeh, C., & Pokay, P. (1994). The new mathematic classroom: Integrating technology and alternative assessments. *Journal of Technology and Teacher Education*, 2 (4), 307-315.

Taylor, R. (Ed.), (1980). *The computer in the school: Tutor, tool, tutee*. New York: Teacher College Press.

Tomlinson, H. & Henderson, W. (1995). Computer supported collaborative learning in schools: A distributed approach. *British Journal of Educational Technology*, 26 (2), 131-140.

Uslick, J., & Walker, C. (1994, April). *An evaluation of an innovation: Standardized test scores were not valid indicators of success*. Paper presented at the annual meeting of the American Educational Research Association, New Orleans, LA. (ERIC Document Reproduction Service No. ED 372 099).

Vaughan, T. (1996). *Multimedia: Making it work* (3rd ed.). Berkeley, CA: Osborne McGraw-Hill.

Wetzel, K. (1996-97). Teacher technology training: Curriculum-based or personal? *Learning and Leading with Technology*, 24 (4), 58-59.

Wharton, C., Rieman, J., Lewis, C., & Polson, P. (1994). The cognitive walkthrough method: A practitioner's guide. In J. Nielsen & R. L. Mack (Eds.), *Usability inspection methods* (pp. 105-140). New York: John Wiley & Sons.

White, J. (1995). Whenever you call... How to provide teachers with training and support when and where they need it. *Electronic Learning*, 15 (3), 16.

Wiburg, K. (1994). Integrating technologies into schools: Why has it been so slow? *The Computing Teacher*, 22 (5) 6-8.

Widmer, C., & Sheffield, L. (1998). Modeling mathematics concepts. *Leading & Learning with Technology*, 25 (5), 32-35.

Wiedenbeck, S., & Zila, P. L. (1997). Hands-on practice in learning to use software: A comparison of exercise, exploration, and combined formats. *ACM Transactions on Computer-Human Interaction*, 4 (2), 169-196.

Withrow, F. B. (1997). Technology in education and the next twenty-five years. *T.H.E. Journal*, 24 (11), 59-61.